

Evaluation of POS Tagging for Web as Corpus

Eugenie Giesbrecht

Institute of Cognitive Science

University of Osnabrück

Supervisors: Dr. Stefan Evert¹, Dr. Marco Baroni²



A thesis submitted for the degree of M.Sc. in Cognitive Science

September 23, 2008

¹University of Osnabrück / Germany, Institute of Cognitive Science

²University of Trento / Italy, Center for Mind and Brain Sciences

Contents

1	Introduction	1
2	State of the Art	4
2.1	Existing Approaches to POS Tagging	5
2.1.1	HMM Taggers	5
2.1.1.1	Introduction to Markov Model Taggers	5
2.1.1.2	General Form of HMM	7
2.1.1.3	Stochastic Parts Program (Church, 1988)	8
2.1.1.4	TreeTagger (Schmid, 1994)	9
2.1.1.5	TnT - Tagger (Brants, 2000)	13
2.1.2	Transformation-based Tagger	15
2.1.3	Maximum Entropy	18
2.1.4	Combining Knowledge-based and Statistical Taggers (Tapanainen and Voutilainen, 1994)	22
2.1.5	Further Tagger Combinations	26
2.1.6	New Trends	30
2.1.6.1	Distributional POS Tagging - Unsupervised and Knowledge-Free	31
2.1.6.2	Support Vector Machines (SVM)	33
2.2	Inter-annotator Agreement	34
2.3	Summary	35
3	Resources, Tools, Methodology	39
3.1	Experiment Setup	39
3.1.1	Tagger Evaluation	39

3.1.2	Tagset Evaluation	41
3.1.3	Text Type Evaluation	44
3.2	Corpora	44
4	Tiger Evaluation	47
4.1	Tagger Evaluation	47
4.1.1	TreeTagger	47
4.1.1.1	Disambiguation Error Statistics for SPF	50
4.1.1.2	Error Analysis	52
4.1.2	Stanford MaxEnt Tagger	55
4.1.3	Tagger Evaluation Results	56
4.2	Tagset Evaluation	58
4.3	Summary	60
5	DeWaC Evaluation	61
5.1	Tagger Evaluation	61
5.1.1	TreeTagger	61
5.1.2	Disambiguation Error Statistics	63
5.1.3	Error Analysis	64
5.1.4	Tagger Comparison	67
5.2	Text Level Evaluation	68
5.3	Tagset Evaluation	69
5.4	Summary	72
6	Conclusions	73
	References	82
	Abbreviations	82

List of Figures

2.1	Decision Tree Example (Schmid, 1995)	10
4.1	Examples of <i>als</i> Annotation in TIGER	53
4.2	Confusion Matrix for Tiger Tagging with a Mapped Tagset	59
5.1	Confusion Matrix for DeWaC Tagging with a Mapped Tagset . . .	70

List of Tables

2.1	Tagging accuracies for TreeTagger with all extensions (Schmid, 1995)	13
2.2	Tagging accuracies from 10-fold cross validation for NEGRA and PENN	14
2.3	Performance on Test Set of the PENN TREEBANK (Brill, 1994) . .	16
2.4	Unsupervised Training: Test Set Accuracy (no unknown words) (Brill, 1995)	17
2.5	Feature Templates for Maximum Entropy Model (Ratnaparkhi, 1996)	19
2.6	Baseline Performance on Development Set (Ratnaparkhi, 1996) . .	20
2.7	Performance of Specialized Model on Test Set (Ratnaparkhi, 1996)	21
2.8	Performance of Baseline and Specialized Model on Consistent Subset of Development Set (Ratnaparkhi, 1996)	22
2.9	Relative Tagger Accuracies (Brill and Wu, 1998)	26
2.10	Complementarity Rates for Taggers' Errors (Brill and Wu, 1998) .	27
2.11	(Volk and Schneider, 1998)	28
2.12	TreeTagger with Extended Lexicon (Volk and Schneider, 1998) . .	29
2.13	test	29
2.14	Individual Tagger Test Set Accuracy (Halteren et al., 2001)	30
2.15	Accuracies in % for SVMTool (Gimenez and Marquez, 2004)	33
2.16	Reported Published Evaluations of POS Taggers for English (Penn Treebank)	37
2.17	Reported Published Evaluations of POS Taggers for German	37
4.1	TreeTagger Evaluation on Tiger	48

LIST OF TABLES

4.2	Disambiguation Error Statistics for TREETAGGER and TIGER Corpus with Standard Parameter File (using after-processing)	50
4.3	Ambiguity Classes Statistics	51
4.4	Disambiguation Error Statistics for TREETAGGER and TIGER Corpus with Standard Parameter File (without after - processing) . .	52
4.5	Tags with the most errors (TIGER, Standard Parameter File) . . .	54
4.6	Most frequent error types	54
4.7	Mostly misclassified words	55
4.8	10-fold Cross Validation for TIGER with Stanford Tagger	56
4.9	Tagger Evaluation on TIGER	57
4.10	TREETAGGER Accuracy for a Mapped Tagset	58
5.1	Tagging accuracies for TREETAGGER on a section of DEWAC vs. TIGER	62
5.2	Disambiguation Error Statistics for TREETAGGER and DEWAC Corpus with Standard Parameter File (using after-processing) . .	64
5.3	Ambiguity Classes Statistics	65
5.4	Tags with the most errors (DEWAC, Standard Parameter File) .	65
5.5	Most frequent error types in DEWAC	66
5.6	Most frequently misclassified words from DEWAC	66
5.7	Tagger Comparison for DEWAC	67
5.8	DEWAC Text Level Accuracies (Total Accuracy and Accuracy for Unknown Words) with Standard Parameter File	68
5.9	Standard Parameter File Accuracy for DEWAC with Mapped Tagset	70
5.10	Standard Parameter File Accuracies for a Mapped Tagset on the Level of Individual Texts	71

Chapter 1

Introduction

Part of Speech Tagging (POS) is the process of assigning a grammatical class marker to each word in a corpus.

POS tagging is a necessary preprocessing step in most, not to say, all NLP applications. The output of part-of-speech taggers is usually forwarded then to parsers. Just to name some of the applications, POS tagging is employed in speech processing, information retrieval and extraction, word-sense disambiguation, corpus annotation projects, and many other tasks (e.g. [Chowdhury and McCabe \(1993\)](#), [Dang and Palmer \(2002\)](#), [Glickman and Jones \(1999\)](#), [Jiménez et al. \(2003\)](#)).

At a first sight, tagging seems to be a rather easy task. It is much easier to solve than parsing, and its accuracy is quite high: 96-97.5% of tokens are assigned a correct part of speech by the most successful approaches ([Schmid \(1995\)](#), [Toutanova et al. \(2003\)](#), [Brants \(2000\)](#)). Thereby it is often considered a "solved task". Though these numbers seem to be quite good, they should be taken with a pinch of salt. 97% of errors in a 350000-token text mean that every 33th token is assigned a wrong part of speech; with accuracy of 96%, every 25th token is misclassified, and almost every sentence contains an error. Furthermore, part-of-speech tagging is just a pre-step to further processing. If the error rate already at this step is so high, then one can easily imagine the increasing propagation of error in further processing.

It seems, however, that part-of-speech (POS) tagging achieved its upper limit - an approximate 97% bound, as predicted by [Church \(1992\)](#), as it is not getting

better since 10 years. Even these 97% should not be taken for granted though as:

- these numbers come from the experiments where the training and test texts come from the same collection, i.e. the same genre and less unknown words than in general case. They are presumably true for *classic* newspaper or fiction corpora. Nowadays we experience the growth of electronically available information, and web as the main source of such information is gaining in importance for both *real-life applications* and *academic research*. The question arises, whether taggers trained on newspaper corpora would perform exactly as good on web texts? No analogous evaluation has been done for web corpora to our knowledge.
- these numbers do not say anything about the performance of individual tags where accuracy for some specific classes is much lower than for the others, the problem which needs detailed examination ([Halteren et al., 2001](#)).

Thereby, the contributions of this thesis are:

1. a thorough evaluation of a mostly used and best reported statistical tagger for German (TREETAGGER) in respect to a traditional newspaper corpus, TIGER corpus ([Brants et al., 2002](#)), which is the biggest available corpus for German at the moment and has not yet been investigated in published part-of-speech tagging research;
2. an evaluation of the TREETAGGER's performance and employability of the standard parameter files for web corpora;
3. a comparison (for both TIGER and web corpus) of the TREETAGGER with the currently best tagger for English - STANFORD TAGGER, and another hidden Markov model based tagger - APACHE UIMA HMM TAGGER, which is an implementation of a state-of-the-art HMM tagger;
4. an evaluation of using a rather small tagset containing basic grammatical categories instead of a fine-tuned one for tagging of both types of corpora.

We start with a survey of the state-of-the-art in part of speech tagging. The employed methodology and corpora used in our experiments are described in Chapter 3. The evaluation of TIGER and DEWAC corpora is reported in Chapters 4 and 5 respectively. In the end, the results of this work are summarized.

Chapter 2

State of the Art

There are essentially two sources of information available to decide on the word’s part of speech in context: syntactic or contextual information, and lexical information.

One of the first known taggers was a deterministic rule-based tagger ([Greene and Rubin, 1971](#)), that was used for initial tagging of the Brown Corpus ([Francis and Kucera, 1982](#)). The algorithm looked up a word in the precompiled general and morphological dictionaries, then the words, ambiguous in their part-of-speech, were disambiguated with the so-called “context-frame rules”. The program achieved only 77% accuracy.

Later it was shown, that just knowing a word and having enough statistics about it one, can achieve a much better performance than with as heap of hand-crafted rules. Thus, a statistical tagger that simply assigns the most common tag to each word gets 90% correct, which is often taken as a “baseline” for taggers ([Charniak et al., 1993](#)).

One can distinguish two major trends in tagger approaches: stochastic and rule-based. The definitional difference between both is very subtle sometimes, as rule-based approaches nowadays tend to use statistical information as well, mostly for training ([Brill, 1995](#)), which implies that if we speak about rule-based techniques these days, it does not exclude that statistics is being employed within the corresponding methods at some point, unlike in early rule-based methods ([Greene and Rubin, 1971](#)). Consequently, when we speak about stochastic taggers, we mean taggers which use stochastic information at the immediate process

of tagging. Stochastic tagging exploits known statistics about the text and, therefore, mostly necessitates a training corpus to get the probability estimates. The parameters of the stochastic models can also be learned in an automatic way, but the results are generally better if a training corpus is present.

In this chapter we describe existing approaches to part-of-speech tagging and go into details of the most influential taggers. In the end, we summarize the performance and the problems of the major taggers.

2.1 Existing Approaches to POS Tagging

2.1.1 HMM Taggers

2.1.1.1 Introduction to Markov Model Taggers

Hidden Markov Models (HMM) are the mainstay of the applications employing statistical modeling in any form, like speech recognition and production systems, signal processing, part of speech tagging.

A Hidden Markov Model is a probabilistic function of a Markov process. A Markov process is a process that fulfills Markov assumptions.

Markov assumptions are:

- limited horizon

Markov processes are states without memory, except for condition of the current state. Though we usually consider sequences of variables that are not independent of each other, it often suffices to know the value of the current situation without going deep into the past happenings. As [Manning and Schütze \(1999\)](#) put it, we do not really need to know how many books were in the library last week or last year in order to predict how many books there will be tomorrow. It is often enough to know the current situation. Thereby, future states in the Markov process are independent of the past, they only depend on the present. Let $X = (X_1, \dots, X_T)$ be a sequence of random variables taking the values from the finite state space $S = s_1, \dots, s_N$, then a limited horizon property could be formalized by:

2.1 Existing Approaches to POS Tagging

$$P(X_{t+1} = s_k | X_1, \dots, X_t) = P(X_{t+1} = s_k | X_t) \quad (2.1)$$

- time invariance

The probabilities do not change over time, i.e. if we know that the probability of observing a rainbow after the rain is equal to 90%, we know that it should be true for today as well as for tomorrow.

If X conforms to these two properties, then it is said to be a Markov chain.

One can describe a Markov chain by a transition matrix:

$$A = a_{i,j} = P(X_{t+1} = s_j | X_t = s_i) \quad (2.2)$$

- with $a_{i,j} \geq 0, \forall i, j$ and $\sum_{j=1}^N a_{i,j} = 1, \forall i$

Additionally we need to specify the probabilities of different initial states for the Markov chain:

$$\pi_i = P(X_1 = s_i) \quad (2.3)$$

- with $\sum_{i=1}^N \pi_i = 1$

Markov models can be used whenever one needs to model the probability of a linear sequence of variables (Manning and Schütze, 1999). Most of the known implementations of the POS taggers are viewing text as being produced by a hidden Markov model, so that tagging can be regarded as a Markov process deciding which states the system went through to generate a given text.

One distinguishes Visible Markov Models (VMM) vs. Hidden Markov Models. The difference is that when we work with "visible" events, we can directly estimate the corresponding probabilities. Thus, we speak about VMM if we are modeling letter sequences or word sequences.

Finding a sequence of part of speech tags in contrast is a different level of abstraction as they are not directly observable. In such cases an additional "hidden" structure is introduced into a VMM that allows us to look at the order of categories of words (Manning and Schütze, 1999).

2.1.1.2 General Form of HMM

A HMM is a five-tuple (S, O, Π, A, B) where:

- S - the set of states (here: parts of speech)
- K - the set of observations (here: words)
- Π - initial state probabilities
- A - state transitions probabilities
- B - symbol emissions probabilities

Furthermore, X_t (state sequence) and O_t (output sequence) are given.

A program for a Markov process (Manning and Schütze, 1999)

1. $t := 1$
2. Start in state s_i with probability π_i (i.e., $X_1 = i$)
3. forever do
4. Move from s_i to s_j with probability $a_{i,j}$ (i.e. $X_{t+1} = j$)
5. Emit observation symbol $o_t = k$ with probability $b_{i,j,k}$
6. $t := T + 1$
7. end

The goal of HMM - based tagger is to find part of speech tags (= hidden states) that generate a sequence of words, that is to look for the most probable sequence of tags underlying the observed words.

Despite their limitations, HMM are one of the most successful techniques in natural language processing and are widely used.

2.1.1.3 Stochastic Parts Program (Church, 1988)

Kenneth Church pioneered an epoch of Hidden Markov Model (HMM) taggers with his *stochastic parts program* (Church, 1988). The programme implemented linear time dynamic programming algorithm, doing basically the same as Viterbi Algorithm, to maximize the product of lexical and contextual probabilities. Lexical probability is the probability of observing part of speech x given word y . Contextual probability in contrast is the probability of observing part of speech x given k previous parts of speech. Both were estimated on the Brown Corpus (Francis and Kucera, 1982).

According to the Zipf's law (Zipf, 1935) however, there will always be a lot of words that are too rare in the corpus, independent of the its size, to reliably estimate the corresponding counts. Church addresses this issue of *data sparseness* by using a very simple "add one" smoothing strategy relying on an external dictionary. This means that if a word is listed in the dictionary with certain possible parts of speech, we add one to the corresponding frequency counts for a given word with a given part of speech tag. In many cases it would not change anything, but on some occasions it can happen that a word can get a different grammatical meaning than in the training corpus (e.g. CANS in the Brown corpus is used only as a noun, but a verbal reading becomes possible when using smoothing). In such cases, "dictionary - smoothing" could be very helpful. The same is true for n-gram counts.

Church's program achieved accuracy of 95%-98%. Most tagging errors of his program were due to "defects in the lexicon", and just few due to the oversimplified grammar (trigram model). The latter implies that "long distance" dependencies are apparently not so important in POS tagging, though they surely contribute a lot to the error percent. Furthermore, Church (1988) claims that numerous examples in the linguistic literature concerning lexical ambiguities, like *Flying planes can be dangerous*, are a little bit far-fetched in the sense that most texts are not that hard in practice. There are only a few "Garden Path"¹ cases.

¹"Garden path sentences" is the term used in psycholinguistics to denote sentences with ambiguous meanings, in order to illustrate the sequential, "one word at a time", processing of language by humans. The most-cited example of such a sentence is "*The horse raced past the barn fell*".

2.1 Existing Approaches to POS Tagging

Generally there is always a "unique" best interpretation which can be found with limited resources, so Church.

Though the programme of Church was very successful for English, it would surely have more problems with morphologically productive languages, like German. Statistical model would end up with a very large number of different word forms, which leads to a huge amount of lexical parameters in such languages. Those are difficult to estimate reliably due to insufficient amount of statistics for them.

Helmut Schmid (Schmid, 1995) and Thorsten Brants (Brants, 2000) recognized that methods were needed to achieve high accuracy with small amounts of training data. Both of them extended independently basic HMM tagger with further subtle details to better deal with morphology-rich languages.

2.1.1.4 TreeTagger (Schmid, 1994)

TreeTagger is a Markov model tagger which employs binary decision trees, whose nodes are guaranteed to be non-zero, instead of smoothing to get reliable estimates for sparse lexical and contextual parameters. Decision trees allow to determine flexibly the size of the context according to some predefined threshold. The context is not just defined by possible trigrams and bigrams, but also by the negative ones: e.g. $tag_{-1} = ADJ$ and $tag_{-2} \neq ADJ$ and $tag_{-2} \neq DET$.

Thus, **TREETAGGER** is using, like a conventional HMM tagger, Markov chains and specifically Viterbi algorithm to find the most probable sequence of tags, but unlike typical n-gram taggers, it is using a binary decision tree which is obtained from the training corpus (see Figure 2.1) to restrict the context to only possible combinations. Analogously, a suffix tree to handle unknown words is constructed as a decision tree.

The highest reported accuracy for the English trigram model, tested on the PENN TREEBANK, within the above described paradigm is 96.34% (Schmid, 1994).

While the results for English with this basic model, where large training corpora are available, were good enough, the results for German with moderate amounts of annotated corpora were less satisfying. This was due to the fact that

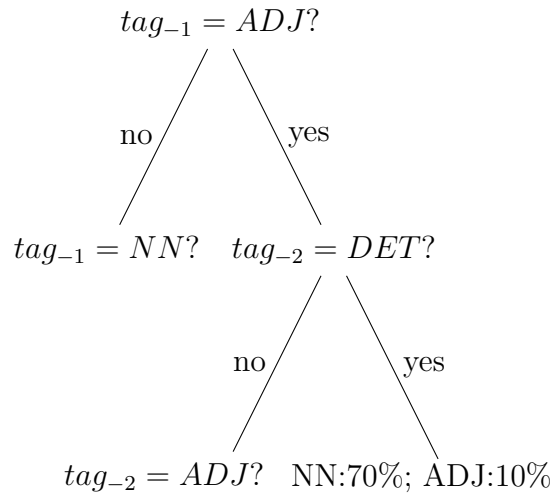


Figure 2.1: Decision Tree Example (Schmid, 1995)

only about 50% of the word forms in the test set had occurred in the training corpus. Consequently, some further modifications of the model were needed to deal with this fact.

Schmid (1995) introduces a couple of new heuristics into the TREETAGGER to deal with poor lexical probability estimates obtained from small corpora. The improvements include:

1. **smoothing with equivalence classes**

If the assumption holds that the words with the same set of possible parts of speech have mostly similar probability distributions, it should be possible to get reasonable probability estimates for rare words from the probabilities of the "equivalently distributed" words. Tag probabilities for frequent words are calculated further on from their corresponding frequencies.

2. **prefix lexicon**

Further assumption is that the influence of the beginning of a word on its POS can be strong in languages, like German, which also have inflectional prefixes. Thus, a prefix lexicon is built in the same way as for suffixes and is combined with the latter.

3. information from automatically tagged corpora

Instead of relying completely on the prefix-suffix estimates to handle unknown words, one could try to enlarge the available lexicon from the automatically tagged corpora. The procedure is the following:

1. Train the tagger
2. Apply trained tagger to the new text in order to get more amounts of (automatically) tagged corpus
3. Extract from the automatically tagged text sets of possible POSs for unknown words
4. Either : - add them to the full form lexicon
or
- calculate probability estimates for such data and combine them with the lexical probabilities in the same way as for the equivalence class probabilities

4. sentence-initial words

Since sentence-initial words are always capitalized, which does not mean that the corresponding word types are always capitalized, it can make sense to look up both variants in the lexicon - capitalized and non-capitalized - for the first words in sentences. Then, if both are present in the dictionary, it is necessary to correspondingly weight them by the relative frequency.

The original tagging formular:

$$P(w_1 w_2 \dots w_n, t_1 t_2 \dots t_n) = \prod_{i=1}^n P(t_i | w_i) / P(t_i) P(t_i | t_{i-k} \dots t_{i-1}) \quad (2.4)$$

- can further be simplified by omitting the factor $1/P(t_i)$, which is justified however only if the a priori probabilities for all tags are about the same:

$$P(w_1 w_2 \dots w_n, t_1 t_2 \dots t_n) = \prod_{i=1}^n P(t_i | w_i) P(t_i | t_{i-k} \dots t_{i-1}) \quad (2.5)$$

The tagger was trained on about 20,000 tokens and tested on 5,000 tokens from the manually annotated part of the German newspaper *Stuttgarter Zeitung*.

2.1 Existing Approaches to POS Tagging

An external lexicon was created from the whole *Stuttgarter Zeitung* by means of extracting all the word forms (36 million words) and analyzing them with DMOR, a German morphological analyzer (Schiller, 1995). All successfully analyzed word forms were stored in the full form lexicon.

The first version of the tagger used the simplified formula 2.5 and the suffix lexicon. It achieved 96.05% accuracy, whereas the original formula 2.4 with all of the above-mentioned extensions resulted in 97.53% accuracy. The average ambiguity for test tokens was 1.45, similarly as for the PENN TREEBANK.

Smoothing with equivalence classes as well as a special treatment of sentence-initial words proved to be most efficient. The gain in accuracy when using a prefix lexicon and the automatically tagged extensions was marginal.

Schmid (1995) analyzed the error sources for German, and detected two major sources of tagger failures:

1. 20% of errors result from interhanging finite and non-finite verb forms. These are so-called *non-local dependency errors* which are impossible to classify correctly with a traditional N-gram tagger. However, it should be possible to catch up such errors with a simple rule-based postprocessor which could be able to learn such transformations automatically, e.g. as described in (Brill, 1995).
2. 18% of errors come from mixing up nouns and proper nouns. Two thirds of such cases are either due to the fact that a proper noun was not in the lexicon of the tagger or it was in the lexicon but not with the right tag (e.g. *Trachtenberg* was only listed as a simple noun). 10% of the problematic cases were proper nouns following an article, i.e. a context where a local tagger strongly statistically prefers a simple noun.

Whereas there was a reduction of 37% in error rate for German with an about 1.5% higher tagging accuracy , the improvements for English were less obvious. Similar tests have been made for English, where two million words from the PENN TREEBANK (Marcus et al., 1994) were used for training and 100,000 unseen tokens for testing. The lexicon was created from the training corpus. The enhancement in tagging accuracy for English was only about 0.5%. Thus, smoothing seems to

2.1 Existing Approaches to POS Tagging

be not that much important for English as there is enough training data available to reliably estimate lexical parameters.

	% of unknown words	known accuracy	unknown accuracy	overall acc.
German newspaper ¹	2%	97.4%	78.0%	97.53%
PENN	–	–	–	96.81%

Table 2.1: Tagging accuracies for TreeTagger with all extensions (Schmid, 1995)

2.1.1.5 TnT - Tagger (Brants, 2000)

Thorsten Brants' TnT-tagger is a typical second order Markov model with transition and output probabilities being estimated from the tagged corpus.

In the first step, individual probabilities are calculated as maximum likelihood probabilities, derived from the relative frequencies:

$$\text{Unigrams : } \hat{P}(tag_3) = \frac{\text{frequency}(tag_3)}{N} \quad (2.6)$$

- with N = total number of tokens in the training corpus

$$\text{Bigrams : } \hat{P}(tag_3|tag_2) = \frac{\text{frequency}(tag_2, tag_3)}{\text{frequency}(tag_2)} \quad (2.7)$$

$$\text{Trigrams : } \hat{P}(tag_3|tag_1, tag_2) = \frac{\text{frequency}(tag_1, tag_2, tag_3)}{\text{frequency}(tag_1, tag_2)} \quad (2.8)$$

$$\text{Lexical : } \hat{P}(word_3|tag_3) = \frac{\text{frequency}(word_3, tag_3)}{\text{frequency}(tag_3)} \quad (2.9)$$

As a second step, contextual frequencies are collected and smoothed by means of linear interpolation of tri-, bi- and unigrams:

$$P(tag_3|tag_1, tag_2) = \lambda_1 \hat{P}(tag_3) + \lambda_2 \hat{P}(tag_3|tag_2) + \lambda_3 \hat{P}((tag_3|tag_1, tag_2)) \quad (2.10)$$

- with $\lambda_1 + \lambda_2 + \lambda_3 = 1$

2.1 Existing Approaches to POS Tagging

The values of λ_1 , λ_2 , λ_3 are estimated by deleted interpolation and are context independent.

Finally, suffix trees are used to compute tag probabilities for words that are not in the lexicon. Suffixes of infrequent words (< 10 occurrences) are used to get the estimates, as they are expected to better reproduce the distribution of unknown words.

Probabilities are smoothed by successive abstraction:

$$P(\text{tag}|l_{n-i+1}, \dots, l_n) = \frac{\hat{P}(\text{tag}|l_{n-i+1}, \dots, l_n) + \theta_i P(\text{tag}|l_{n-i}, \dots, l_n)}{1 + \theta_i} \quad (2.11)$$

Furthermore, [Brants \(2000\)](#) maintains two different suffix trees depending on capitalization, which seems to improve accuracy for English, as it capitalizes only proper names, so that one can more or less reliably define a proper tag for a capitalized word.

In the end, 10-fold cross validation on the Negra corpus ¹ as well as on the PENN TREEBANK ([Marcus et al., 1994](#)) is performed. The results are in Table 2.2.

	% of unknown words	known		unknown		overall	
		acc.	σ	acc.	σ	acc.	σ
NEGRA	11.9%	97.7%	0.23	89.0%	0.72	96.7%	0.29
PENN	2.9%	97.0%	0.15	85.5%	0.69	96.7%	0.15

Table 2.2: Tagging accuracies from 10-fold cross validation for NEGRA and PENN

The results in Table 2.2 show that the performance of TNT TAGGER on the PENN TREEBANK is almost the same as that of the TREETAGGER. The accuracies for German cannot be directly compared as the tests have been performed on different corpora and the percentage of unknown words in the case of TNT has been significantly higher. Still the difference between the outcomes is minimal (less than one percent).

¹<http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html>

2.1.2 Transformation-based Tagger

The idea behind the so-called "transformation-based learning" (TBL) is *guess first, and change your mind if necessary* (Brill, 1994, 1995).

TBL is a combination of rule-based and stochastic methodologies. Like rule-based methods, it uses rules to specify tags in certain contexts. Like stochastic tagging, it uses statistics and machine learning for learning the rules. Brill's tagger is based on the transformation-based error-driven learning algorithm, which is widely used in various areas of natural language processing e.g. Ramshaw and Marcus, Samuel and Vijay-Shanker (1998).

Transformation-Based Error-Driven Learning

- 1 -Pass unannotated text through an initial-state annotator
- 2 -Compare the output to the "truth" (manually annotated corpus)
- 3 -From step (2) learn good transformations
- 4 that make output better resemble the "truth"
- 5 -Add a best learned rule to the ordered list of transformations.

Before the learning can begin, one has to specify the following:

1. the initial state annotator,
2. the space of allowable transformations,
3. the scoring function for comparing the corpus to the *truth* and choosing a transformation.

Once the transformations are learned, tagging consists in annotating a new text by going through the ordered list of transformations.

In Brill (1994), the above-mentioned specifications are following:

2.1 Existing Approaches to POS Tagging

1. all words are initially tagged with their most likely tag, statistics is obtained from the training corpus,
2. 21 transformations templates are used (e.g. *The preceding or following word is tagged y*),
3. the scoring function is tagging accuracy, so that a transformation is learned which results in the greatest error reduction.

An example of a deduced rule could be: *"Change the tag of a word from VERB to NOUN if the tag of the following word is VERB"*.

Brill (1994) claims that a rule-based tagger can achieve competitive performance with stochastic taggers. He evaluates his tagger on the PENN TREEBANK. The first 950,000 tokens of the corpus are used for training and the next 150,000 for testing. 148 rules for tagging of unknown words and 267 contextual rules have been learned.

total acc.	unknown acc.
96.50%	85.00%

Table 2.3: Performance on Test Set of the PENN TREEBANK (Brill, 1994)

In order to avoid the dependence of the taggers on the large amount of manually annotated text, Brill (1995) proposes an unsupervised and a semi-supervised variants of TBL tagger where only a dictionary of possible tags for each word is available.

1. The initial state annotator tags all words in the corpus with all allowable tags.
2. Four transformations are allowed, which are taking into consideration previous and next word or tag. Here transformations are reducing uncertainty and disambiguating a tag, instead of changing one tag into another as in supervised learning. E.g., *Change the tag from NN_VB_VBP to VBP if the previous tag is NNS*.

2.1 Existing Approaches to POS Tagging

3. A scoring function is computed based on current state of tagging. A good transformation in this case is the one for which one of the possible tags appears more frequently with unambiguously tagged words in the same context.

Then, the experiments on two corpora and tagsets are run: the PENN TREE-BANK (Marcus et al., 1994) and the original BROWN CORPUS (Francis and Kucera, 1982). The test set contains **no unknown words**. In the PENN TREE-BANK experiments, a training set of 120,000 words and a test set of 200,000 words is used. After learning 1,151 transformations, the training set accuracy goes up from 90.7% (initial state tagging) up to 95.0 % (95.1% for the test set). The fact that both - training and test set accuracies - go hand in hand demonstrates that no overtraining occurs. The similar result is obtained with the BROWN CORPUS as well. Both tests are summarized in the table 2.4.

Corpus	Training Size (Words)	Accuracy (no unknown words!)
PENN TREEBANK	120K	95.1%
Brown Corpus	120K	95.6%
Brown Corpus	350K	96.0%

Table 2.4: Unsupervised Training: Test Set Accuracy (no unknown words) (Brill, 1995)

In the semi-supervised variant, untagged text is first sent to the unsupervised initial state annotator where unsupervised transformations are learned as afore described. Then the latter are applied to a new text, which is then passed to the supervised learner along with the manually annotated corpus, and a supervised learner learns second-order transformations.

In all the tests, the tagger using semi-supervised learning has outperformed a purely supervised performance at basically no cost in terms of additional manual annotation.

In this paper, Brill ignores however the problem of unknown words which is crucial for tagging performance, so it is difficult to estimate the *true performance* of this method.

2.1.3 Maximum Entropy

Maximum entropy (MaxEnt) modeling is a probabilistic framework where a model is found which is both consistent with the observed data and is maximally agnostic with respect to all parameters for which no data exist. The goal is to maximize the entropy¹ of a distribution, which is at the same time subject to certain constraints. The philosophy behind maximum entropy models is to maximize uncertainty for unknown features in order to avoid overfitting. The pioneer of MaxEnt modeling for POS tagging is Ratnaparkhi (1996). MaxEnt taggers combine the advantages of a rich feature representation, like *rule-based* taggers (Brill, 1995), and at the same time uses the power of statistics, as *Markov model* taggers (Brants, 2000; Church, 1988).

The probability model within MaxEnt framework is defined as:

$$p(h, t) = \pi \mu \prod_{j=1}^k \alpha_j^{f_j(h, t)} \quad (2.12)$$

- where π is a normalization and $\mu, \alpha_1, \dots, \alpha_k$ are the positive model parameters and f_1, \dots, f_k are known as *features* with $f_j \in 0, 1$.

Given a sequence of words w_1, \dots, w_n and tags t_1, \dots, t_n , define h_i as the history available when predicting t_i . The parameters $\mu, \alpha_1, \dots, \alpha_k$ are chosen to maximize the likelihood of the training data using p :

$$L(p) = \prod_{i=1}^n p(h_i, t_i) = \prod_{i=1}^n \pi \mu \prod_{j=1}^k \alpha_j^{f_j(h_i, t_i)} \quad (2.13)$$

The entropy of such distribution is defined as following:

$$H(p) = - \sum_{h \in H, t \in T} p(h, t) \log p(h, t) \quad (2.14)$$

Thus, the goal within the Maximum Entropy formalism is to maximize the entropy of the above distribution subject to certain constraints, defined by:

$$E f_j = \tilde{E} f_j, 1 \leq j \leq k \quad (2.15)$$

¹Entropy is a numerical measure of uncertainty of the outcome

2.1 Existing Approaches to POS Tagging

- where the model's feature expectation is:

$$Ef_j = \sum_{h \in H, t \in T} p(h, t) f_j(h, t) \quad (2.16)$$

- and the observed feature expectation is:

$$\tilde{E}f_j = \sum_{i=1}^n \tilde{p}(h_i, t_i) f_j(h_i, t_i) \quad (2.17)$$

- where $\tilde{p}(h_i, t_i)$ denotes the observed probability of (h_i, t_i) in the training data.

The joint probability of a history and a tag is defined by those features which are active. Features can virtually encode any information one considers useful. After the set of features is defined, a model is generated from the training corpus by scanning each pair (h_i, t_i) in the training corpus with *feature templates* (see Table 2.5). Thus, a history can be defined as in the following way:

$$h_i = w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}$$

Condition	Features
w_i is not rare	$w_i = X$ $t_i = T$
w_i is rare	X is prefix of w_i , $ X \leq 4$ $t_i = T$
	X is suffix of w_i , $ X \leq 4$ $t_i = T$
	w_i contains number $t_i = T$
	w_i contains uppercase character $t_i = T$
	w_i contains hyphen $t_i = T$
$\forall w_i$	$t_{i-1} = X$ $t_i = T$
	$t_{i-2}t_{i-1} = XY$ $t_i = T$
	$w_{i-1} = X$ $t_i = T$
	$w_{i-2} = X$ $t_i = T$
	$w_{i+1} = X$ $t_i = T$
	$w_{i+2} = X$ $t_i = T$

Table 2.5: Feature Templates for Maximum Entropy Model (Ratnaparkhi, 1996)

The generation of parameters for unknown words is based on the same assumption as in Brants (2000), that rare words in the corpus are more predictive

2.1 Existing Approaches to POS Tagging

for the unknown features than the frequent ones (Ratnaparkhi, 1996). A simple cut-off is used for features which occur less than ten times in the corpus and consequently are considered as unreliable.

The tagger proceeds as following: let $W = w_1, \dots, w_n$ be a test sentence and let t_{ij} be the j th highest probability tag sequence up to and including word w_i .

Max Ent

1. For $i = 1$: generate tags for w_1 ,
find the most probable N tags up to now and
set t_{1N} correspondingly, with $1 \leq j \leq N$.
 2. For $i = 2$: (a) Initialize $j = 1$
(b) While $j \leq N$: Generate tags for w_i and append them to $t_{(i-1)j}$
(c) $j = j + 1$, Go to (b)
 3. Find N highest probability sequences generated by (2) and
set t_{ij} for $1 \leq j \leq N$ accordingly.
 4. $i = i + 1$, Repeat from (a) if $i \leq n$
 5. Return highest probability sequency t_{n1}
-

In addition the search procedure can also consult a *tag dictionary*, which lists for each known word its possible parts of speech.

The experiments were run on the Wall Street Journal data. 962687 tokens of text were used for training, 192826 for development and 133805 for testing.

	% of unknowns	total acc.	unknown acc.	sentence acc.
Tag Dictionary	3%	96.43%	86.23%	47.55%
No Tag Dictionary	3%	96.31%	86.28%	47.38%

Table 2.6: Baseline Performance on Development Set (Ratnaparkhi, 1996)

The running time of the parameter estimation within this framework is $O(NTA)$ where N is the training set size, T is the number of allowable tags and A is the average number of features that are active for given tags. The running time of a search procedure on a sentence of length N is $O(NTAB)$ where N is the size of

2.1 Existing Approaches to POS Tagging

% of unknowns	total acc.	unknown acc.	sentence acc.
2.65%	96.63%	85.56%	47.51%

Table 2.7: Performance of Specialized Model on Test Set (Ratnaparkhi, 1996)

the sentence, T and A are defined above, and B is the beam size¹. Under such conditions, the method is almost inapplicable for online applications.

The advantage of maximum entropy models, however, is that one can integrate virtually any amount of relevant features, which could be useful for some "difficult" cases where the general feature statistics fails. Words that were mistagged at least 50 times when the training set was tagged with the baseline model were defined arbitrarily as "difficult" (e.g. ABOUT and THAT).

The assumption that a better modeling of context features should improve tagging, if the errors were indeed due to the insufficient feature modeling, was tested. Specialized feature set for these cases was constructed using the baseline context and the information about the identity of the current word itself. An example of the specialized feature for the word ABOUT could be:

$$f_j(h_i, t_i) = \begin{cases} 1, & \text{if } w_i = \text{about} \ \& \ t_{i-1}t_{i-1} = \text{DT NNS} \ \& \ t_i = \text{IN} \\ 0, & \text{otherwise} \end{cases}$$

The model performed with 96.49% accuracy on the Development Set with a special treatment for "difficult" cases, thereby just insignificant improvement was observed. For some words, like ABOUT the performance had even gone down. Ratnaparkhi (1996) detected that the tagging distributions of ABOUT or AGO changed precisely where the annotator was changed. This hypothesis was tested in that training and testing was done on the subpart of the corpus which had been annotated by one person, and the accuracy indeed got higher (see Table 2.8).

Obviously, there are *inter-annotator* disagreements in the PENN TREEBANK corpus. The improvement by using the specialized model has still been too modest though. The latter implies either that the used features needs further improvement or that the *intra-annotator* inconsistencies exist in the PENN TREEBANK.

¹"Beam search" is used here to find candidate tag sequences.

2.1 Existing Approaches to POS Tagging

Training Set (words)	Test Set (words)	Baseline	Specialized
571190	44478	97.04%	97.13%

Table 2.8: Performance of Baseline and Specialized Model on Consistent Subset of Development Set (Ratnaparkhi, 1996)

In Toutanova et al. (2003), they extend successful MaxEnt models by introducing:

- a bidirectional inference into the model, i.e. taking into consideration not exclusively either preceding or following tags but both;
- and more lexicalization, i.e. they use information about preceding and following words together with the information about their corresponding parts of speech.

When combining both word and tag sequence features, they achieve 97.24% accuracy (and 89.04% for unknown words) on the Test Set of the Penn Treebank, which is the best reported accuracy for "singleton" English taggers.

2.1.4 Combining Knowledge-based and Statistical Taggers (Tapanainen and Voutilainen, 1994)

As pointed out by Brill and Wu (1998), the errors, most state-of-the-art taggers do, are highly complementary. This caused quite a number of experiments with tagger combinations, as the complementarity of the errors seemed to imply that there is sufficient information available, but that this or that algorithm fails to apply it correctly in some cases (MacKinlay and Baldwin, 2005). Since the nineties a significant amount of research on classifier combinations has been carried on.

Tapanainen and Voutilainen (1994) use a knowledge-based tagger ENGCG (Karlsson, 1995) and a Xerox HMM tagger (Cutting et al., 1992) (the latter is trained in an unsupervised manner), and combine the outputs of both to produce a fully disambiguated text. The main claim of the paper is that a reasonable combination of statistics and rules can achieve almost perfect results, as the errors of statistical taggers that are very naive from the linguistic point of view can be resolved reliably by linguistic rules with next to no errors.

2.1 Existing Approaches to POS Tagging

The procedure is the following:

1. let both taggers do the job,
2. if there is a discrepancy between the outputs: trust the rule-based tagger,
3. if a rule-based tagger leaves an unresolved ambiguity: select the alternative which is closest to the selection made by statistical system.

The system was tested on 26,711 words from the *Wall Street Journal*, the *Economist* and *Today*, all taken from the BANK OF ENGLISH CORPUS (Järvinen, 1994). The tagsets used by both system were different and needed to be aligned. ENGCG tagset consisted of 180 tags and Xerox tagger had been trained on the Brown Corpus (Francis and Kucera, 1982) and had 94 tags. The mapping was not always straightforward, especially in the cases where the text was differently tokenized by the taggers.

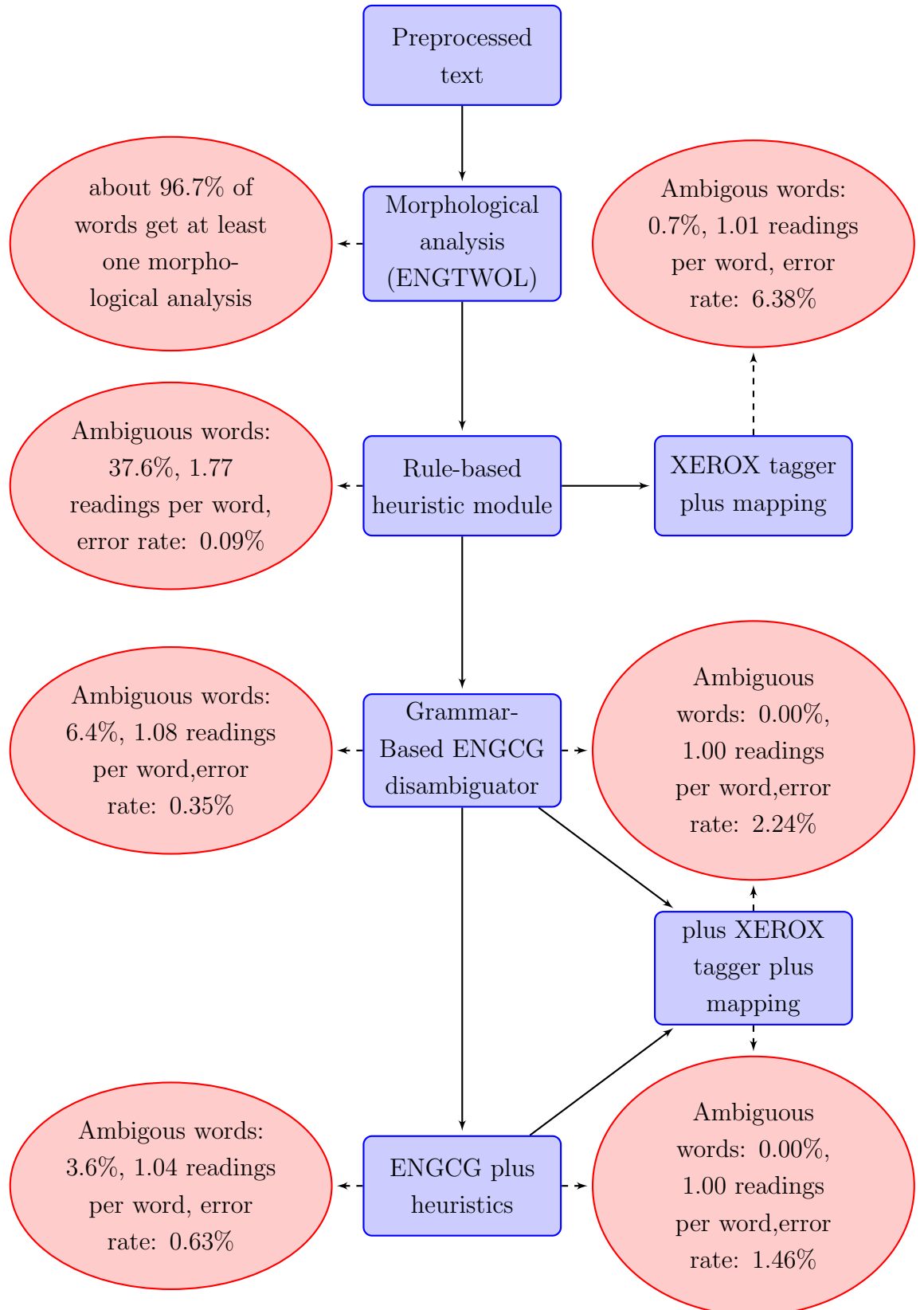
The tagging accuracy after combining linguistic and statistical components was over 98.5%. It is the *highest reported* accuracy for English tagging.

Diagram 2.1.4 shows the tests with their corresponding accuracies in Tapanainen and Voutilainen (1994). It is easy to follow the error propagation with more and more components being involved on the basis of this diagram. The error rate after morphological analysis was 0.9%; after the ENGCG disambiguation with heuristic constraints the error was as high as 0.63% and 57% of errors were made with HMM tagger applied after the rule-based one. This fact is not surprising as ENGCG tackled only the structurally "easy" cases and left the "hard" ones for the Xerox tagger. What is quite revealing is that more than a half of the errors made by probabilistic tagger are treated correctly by a carefully designed linguistic rule-based tagger. Obviously, a lot of errors a probabilistic tagger makes are structurally "naive". The accuracy reduction from 98.54% to 97.76% when the Xerox tagger was used without the ENGCG heuristics demonstrates that the ambiguity should be resolved as far as possible with rules. The question of how far one can go using only linguistic constraints stays open in Tapanainen and Voutilainen (1994), but apparently more linguistics knowledge, especially integration of syntactic information, should boost the tagger accuracy.

2.1 Existing Approaches to POS Tagging

It could have been useful to get an error analysis of the cases which could not be reliably disambiguated with a rule-based tagger or with a statistical tagger. Only one example of such cases is mentioned in the paper, namely the confusion between CS (subordinating conjunction) and PREP or IN (preposition). These cases are potential candidates for the tagset modifications.

2.1 Existing Approaches to POS Tagging



2.1.5 Further Tagger Combinations

A couple of years later some further attempts were undertaken to combine statistical and rule-based taggers, however, with less success than Tapanainen and Voutilainen (1994).

In Brill and Wu (1998), the performance of the major state-of-the-art taggers is compared in respect to the *Wall Street Journal (WSJ)*. The corpus is divided into 80% training and 20% testing, which results in about 1.1 million words of training data and 265,000 words of test data. The outcome of the taggers is summarized in Table 2.9, where numbers in parentheses denote ambiguous plus unknown words' accuracies. Unfortunately, the percent of unknown words as well as of ambiguous ones is not reported here, but it is probably more or less the same as in other WSJ experiments, at least for the unknowns (i.e. not more than 2%-3%).

Tagger	Trigram	TBL	MaxEnt
Accuracy (%)	96.36 (93.8)	96.61 (94.3)	96.83 (94.7)

Table 2.9: Relative Tagger Accuracies (Brill and Wu, 1998)

Further, Brill and Wu (1998) compute the complementarity of the errors of different taggers, which is roughly a numeric value of how often one tagger is wrong when another one is right (see Table 2.10). For example, when a *Trigram tagger* is wrong, a *MaxEnt tagger* is right in 42% of cases.

The numbers in Table 2.10 encourage the idea that the necessary information is there and it just cannot be interpreted to a 100% by a tagger. If an oracle could choose between the outputs of the given three taggers, the error rate would decrease to 1.62%, which would result, for example, in 48.8% error reduction over the *MaxEnt tagger* error rate. Brill and Wu (1998) use this fact to show that complementarity is additive, so that the more taggers we combine, the better should be the outcome.

One further important point is that half of the errors occur in the cases where the taggers disagree on the tag. Thus, the error rate for a *Trigram tagger* goes up from 3.64% up to 54.9% when there is disagreement with *MaxEnt* and *TBL* on a proposed tag for a given word. The question would be then, which tagger to trust

2.1 Existing Approaches to POS Tagging

	Trigram	TBL	MaxEnt
Trigram	0	34.6	33.5
TBL	39.0	0	37.7
MaxEnt	42.0	41.7	0

Table 2.10: Complementarity Rates for Taggers' Errors (Brill and Wu, 1998)

in what situations. The simplest strategy to combine the taggers is to use the so-called *simple voting*: the tag that is proposed by the majority of the taggers is chosen, with *MaxEnt* being a favorite in the instances where all disagree. This results in 6.9% error reduction for *MaxEnt* tagger (from 3.2% to 3.0%).

A more sophisticated strategy is to make the choice of the tag or tagger depending on the context, i.e. what Brill and Wu (1998) call *contextual clues*. They use the previous, current and next words and the corresponding tagger outputs as contextual clues. Then two experiments are conducted with these settings:

1. The most likely tag from the training set is chosen, thereby allowing for tags which are not suggested by current tagging outputs, e.g. if a *Trigram tagger* thinks it should be *X* while a *MaxEnt tagger* claims it should be *Y*, then it is often the case that the tag in reality is *Z*. Such scenarios are realistic with this modeling. This resulted in error reduction of 9.8% (from 3.2.% to 2.9%).
2. In the second experiment setting, one defines which tagger to trust more in which situations instead of getting the supposedly right tag from the model. The latter caused an error decrease of over 10.4% (from 3.2% to 2.8%)

The findings of Brill and Wu (1998) may be statistically significant, but they are very encouraging. Apart from a limited applicability of such combinations for online applications, the error reduction was not as much as expected. Instead of error decrease of over 40%, the best obtained reduction for *MaxEnt tagger* was only 10%. Thus, either a hypotheses fails or the tested tagger combinations have not been optimal and further insights are needed.

2.1 Existing Approaches to POS Tagging

Paralelly to Brill and Wu (1998), Volk and Schneider (1998) experiment with a statistical (TreeTagger) and a rule-based (Brill) tagger for German. Both taggers are trained and tested on the same manually annotated newspaper¹ corpus: 60,710 tokens are used for training and 8,887 for testing, where test sentences are from all over the corpus (every eighth sentence). The ambiguity rate in the test corpus is 1.50 tags per word.

	% unknowns	accuracy (unknown)	lex. errors	disambig. errors
TreeTagger	15	95.27% (84.05%)	3.24%	1.49%
TBL	15	94.75% (81.52%)	3.62%	1.63%

Table 2.11: (Volk and Schneider, 1998)

Volk and Schneider (1998) report a rather detailed error analysis, differentiating between lexical and disambiguation errors (see Table 2.11). *Lexical errors* occur when a correct tag is not present in the lexicon, i.e. either a word is unknown or a right tag for a given word is not in the lexicon. In contrast, in *disambiguation errors* a correct tag is in the lexicon but the tagger fails to pick it.

Volk and Schneider (1998) claim on the basis of the outcomes of single taggers that:

1. the TREETAGGER is better at tagging unknown words;
2. BRILL TAGGER is better at disambiguating many-way-ambiguous tokens and special symbols.

In the next step they try to improve the accuracy of single taggers by integrating external lexicon. After having performed morphological analysis of unknown words with Gertwol system (Oy, 1994), all the analyzed word forms with respective tags are added to the tagger lexicon. Though only 109 tokens remain unanalyzed, i.e. the tagger starts with almost complete lexicon, the accuracy increase is just 0.5% for BRILL TAGGER.

The reason is obvious: Gertwol increases the average ambiguity of tokens. In spite of the hypotheses that TBL tagger is better at disambiguating, it seems not

¹Frankfurter Rundschau

2.1 Existing Approaches to POS Tagging

to be very successful with a lexicon extended through morphological analysis. The ordering of tags according to the general probability mass of single tags improves the outcome by only 0.2%, with the best result for combining BRILL TAGGER with Gertwol being **95.45%**. Extending the lexicon of the TREETAGGER improves its accuracy by 1% (from 95.27% to 96.29%, see table 2.12).

	% unknowns	accuracy (unknown)	lex. errors	disambig. errors
TreeTagger	1.23	96.29% (66.06%)	1.65%	2.06%

Table 2.12: TreeTagger with Extended Lexicon (Volk and Schneider, 1998)

We should keep in mind, that these results again are for the same type of text as the training text itself.

In general, major sources of errors for German are the same as reported in Schmid (1994): both taggers have problems with the NN—NE (noun—proper noun) distinction, non-local dependencies as in VVFIN—VVINF (finite vs non-finite verb) and in some recurrent situations with adjectives. We will see, that these kinds of errors are typical for all statistical and non-statistical taggers using only *local* context. Sequentializing of both taggers didn't bring over any considerable improvement; vice versa it decreased the TREETAGGER performance.

Halteren et al. (2001) experiment on different strategies to combine the output of the taggers (see Table 2.13 for an overview of combination strategies).

Voting	<i>Simple</i> : each component classifier gets an equal vote; tag which gets the most votes is selected.
	<i>Weighted</i> : each component classifier's vote is weighted by its accuracy
Stacking	a classifier is trained to predict the correct output class when given as input the outputs of the ensemble classifiers, and possibly additional information. <i>Arbiter effect</i>

Table 2.13: test

Voting techniques cannot suggest a tag that was not "fired" by any of the component taggers, i.e. on the one side, all the taggers are restricted to the same tagset and on the other side, if the right tag was not found by any of the

2.1 Existing Approaches to POS Tagging

taggers, then there is no chance to get the right tag from their combination. In contrast, *stacking* (what Brill and Wu (1998) call *contextual clues*) allows it. The experiments in Halteren et al. (2001) are conducted on three corpora: *Lancaster-Oslo/Bergen corpus* of the British English (LOB, Johansson (1986)), *Wall Street Journal (WSJ)* corpus of the American English and a small *Dutch corpus*. We are interested primarily in the results for the *Wall Street Journal* as it allows us to compare them to other published figures.

On the other side, the outcome of the LOB corpus is also important to look at, because LOB corpus is considered to be more accurately tagged and is more consistent than the WSJ. Furthermore, the LOB tagset contains 170 tags, which introduces more ambiguity (2.82 tags per word), whereas the WSJ corpus is tagged with the PENN TREEBANK tagset that consists of 48 tags (2.34 tags per word). The training/test set separation for both corpora was at the utterance level, like in Volk and Schneider (1998), which means that test and training sentences come from the same texts but test set contains unseen sentences.

	% of unknowns	TBL	MaxEnt	HMM
WSJ	2.3%	96.28%	96.88%	96.63%
LOB	2.51%	96.37%	97.52%	97.55%

Table 2.14: Individual Tagger Test Set Accuracy (Halteren et al., 2001)

As we see from Table 2.1.5, TBL is the weakest in performance. In other respects, the results for tagger combinations in Halteren et al. (2001) are more or less the same as in Brill and Wu (1998), at least for the *Wall Street Journal*, independent of different context weighting and seemingly different stacking strategies. The error reduction for the WSJ from the best tagger combination is just 11.3% - too few to be a promising direction, especially taking into consideration bigger processing times for tagger combinations in comparison to individual taggers.

2.1.6 New Trends

There are also a couple of quite different approaches in the air since the 90-ties that are worth mentioning briefly here, as they present a completely different

methodology in contrast to all the experiments considered till now. These new trends are either purely distributional or classical machine learning approaches.

2.1.6.1 Distributional POS Tagging - Unsupervised and Knowledge-Free

POS tagging is a kind of word - sense disambiguation, in the sense that a POS tag gives some information about the sense of the word *in the context of use*.

Thus, it could make sense to try distributional approaches broadly applied to word-sense disambiguation for part of speech tagging, the idea advocated by Schütze (1995).

Distributional methods originate from Harris's distributional hypothesis (Harris, 1968) that attests that *a word is known by the company it keeps*. Here we classify WORDS based on their distributions and not individual TOKENS. The problem with such procedure is a wide-spread part-of-speech ambiguity of words.

A working hypothesis is that syntactic behaviour is reflected in cooccurrence patterns. The procedure in the approach of Schütze (1995) is the following:

- Get the right and the left context vectors for every word. Vectors have 250 entries corresponding to 250 most frequent words in the Brown Corpus.
- Apply singular vector decomposition (SVD) to avoid data sparseness.
- Left and right context vectors are used as basis for 4 different tag induction experiments:
 1. Induction based on word type only (baseline): concatenated vectors for all 47,025 surface words from the Brown Corpus are formed; SVD reduces the original 500 dimensions of the matrix to 50; the resulting vectors are clustered into 200 clusters; all occurrences of the word are assigned to one class.
 2. Induction based on word type and immediate context - an occurrence of word is represented by a concatenation of 4 vectors: the right context vector of the preceding word, the left and right context vectors of the word itself and the left context vector of the following word. Hereby

2.1 Existing Approaches to POS Tagging

20,000 word triples are randomly selected; dimensions are reduced from 1,000 to 50 by means of SVD; the resulting vectors are clustered into 200 classes (here: *tags*), defined by the centroid of each class. To tag a token: compute its correlation to 200 cluster centroids and then assign the occurrence to the closest cluster.

Problem: this procedure fails on tokens whose neighbours are punctuation marks, that is why the next experiment is settled.

3. The same as 2 but restricted to *natural contexts*, i.e. tokens next to punctuation marks as well as tokens having rare words as neighbours are avoided.
4. Induction based on word type and context using generalized left and right contexts.

The idea behind it is that words with similar left or right contexts characterize words to their right or to their left in a similar way, e.g. *seemed* and *would* have similar left contexts. This concept is similar to *equivalence class smoothing* in Schmid (1995).

Evaluation strategy is not quite clear for distributional tagging. What Schütze (1995) employs is mapping of PENN TREEBANK tags to 16 basic tag categories. Penn Treebank parses of the BROWN CORPUS are used for the experiments. Punctuation marks, special symbols, interjections, foreign words and tags with fewer than 100 occurrences are excluded from evaluation, which makes this method already unsuitable for tagging the web.

The average accuracy in the above-mentioned experiments was for: (1) - 0.53%; (2) - 0.78%; (3) - 0.83%; (4) - 0.78%.

On examination, three sources of errors were found:

1. rare words and rare syntactic phenomena
2. indistinguishable distribution, e.g. VBN(past participle) vs PRD(predicative adjective) are both used mostly as compliments of *to be*.
3. non-local dependencies

2.1 Existing Approaches to POS Tagging

Thus, we see that this method has basically the same problems as probabilistic tagging and it is actually using the same contextual features as other taggers.

The question is, why one would want to apply such a rather poorly performing method, if any semi-supervised algorithm, like HMM with training corpus, is performing better. The primary target is, as [Schütze \(1995\)](#) puts it, rare languages and languages or situations where a pre-tagged corpora are not available.

[Biemann \(2006b\)](#) takes on the idea of distributional tagging and develops it further in that he gives up the idea of the reasonableness of pure vector space models (as in [Schütze \(1995\)](#)), which are not suited to highly skewed distributions common for natural language as well as computationally expensive. Instead, he switches to graph models that are very efficient and have neither dimensionality reduction nor smoothing problems as *zero* entries do not exist in the graphs. One further difference between both approaches is a different clustering algorithm: while [Schütze \(1995\)](#) is using the cosine similarity and Buckshot clustering¹ to find word classes, [Biemann \(2006b\)](#) uses the Chinese Whispers (CW) graph-clustering algorithm ([Biemann, 2006a](#)), that allows finding the number of classes in an unguided way instead of passing it as a parameter to the algorithm.

2.1.6.2 Support Vector Machines (SVM)

Last but not least, a purely machine learning approach to word class tagging is worth mentioning here. [Gimenez and Marquez \(2004\)](#) apply Support Vector Machines (SVMs) ([Christianini and Shawe-Taylor, 2000](#)) to tagging. SVMs allow to integrate virtually any number of features without loss of performance, except that training takes a lot of hours. The accuracies for the WSJ are listed in table 2.15 and show a very competitive performance to the state-of-the-art taggers, even better than most of the reported accuracies till now for the WSJ, except for [Toutanova et al. \(2003\)](#) (97.24%).

	total	known	unknown	ambiguous
SVMTool	97.16	97.39	89.01	93.91

Table 2.15: Accuracies in % for SVMTool ([Gimenez and Marquez, 2004](#))

¹Buckshot clustering is a combination of hierarchical agglomerative clustering and k-means.

2.2 Inter-annotator Agreement

All of the above mentioned approaches rely on training data in order to get a reliable model. These training text corpora are annotated manually, in ideal case, by specially trained for the task professionals and by at least two people annotating the same material. The latter means that two annotators independently of each other tag the same text collection. An intuitive measure of tagging consistency of such a corpus is a rate of disagreement between annotators, expressed as a percentage of the raw number of such disagreements over the total number of manually tagged or corrected tokens (Marcus et al., 1994).

The mean inter-annotator disagreement in the PENN TREEBANK (Marcus et al., 1994) was 4.1% (with a median of 3.6%), whereas a greater proportion of disagreement was due to one text containing a lot of chemical and other formulas. As there was no explicit guideline in respect to tagging of such cases, the annotators tagged such symbols differently. After exclusion of a given text from the annotation task, the mean disagreement drops to 3.5% with the median unchanged at 3.6%.¹

Brants (2000) reports an initial inter-annotator agreement of 98.57% for the NEGRA corpus, which has been improved up to 98.80% after discussing the disagreement cases in a group, but even then it has not got 100%.

On the other side, Tapanainen and Voutilainen (1994) report inter-annotator agreement of over 99% and at close examination they have figured out that the errors in almost all cases had happened due to inattention. Only in few examples, mostly in headings, it has been agreed that a multiple choice could be appropriate due to the ambiguity of the utterance as some of the grammatical information is omitted there. Still the claim here is, a 100% consensus is possible if the analysis is done by the *experts in the agreed grammatical representation with emphasis on the quality of the work* and at least at the level of morphological analysis.

These numbers imply that:

¹Median is a more correct approximation in the case of four annotators as it was in Marcus et al. (1994), as median is better approximating a kind of an average amount of 'errors' made by each of the annotators in respect to other annotators' results.

- taggers are trained on imperfect corpora, which is not that dramatic as they should be resistant to noise;
- taggers are tested on faulty collections which makes a 100% accuracy rather unrealistic. We would not know who was right in certain situations - the tagger or the *gold standard*;
- there are cases where ambiguous tag assignments are possible and where it is not straightforward even for humans which one to choose.

2.3 Summary

State-of-the-art approaches in POS tagging have been presented in this section. Tables 2.16 and 2.17 summarize the corresponding reported accuracies for English and German taggers. The outcomes demonstrate that the assumption of Church (1992) on the upper boundary of tagging seems to be legitimate, i.e. a hundred percent accuracy for taggers looks like an unrealistic goal. Published accuracies range between 95% and ca. 97%-98% since 1988, and are not getting better.

Let us summarize the problems that the taggers are *unable* to deal with properly. As precisely noted by MacKinlay and Baldwin (2005), these problematic 2%-3% percent include:

1. words which cannot be tagged correctly using only local dependencies,
2. errors in the reference corpus (Halteren et al., 2001; Padro and Marquez, 1998; Ratnaparkhi, 1996), which make the evaluation of the tagger's true accuracy almost impossible,
3. and words for which even human annotators have difficulties agreeing on a tag.

As to the first problem, it is due to the algorithms deficiencies, as almost all of them use more or less the same *features* that model only local context, except for the approach of Tapanainen and Voutilainen (1994). We will not dwell on this issue further as general improvements of the algorithms are out of the scope of this thesis.

Problems (2) and (3) are interconnected in a way, as most inconsistencies in reference corpora occur in such cases where human annotators themselves have difficulties, and as we've seen in Section 2.2, inter-annotator agreement in *gold standard* corpora leaves much to be desired.

The intuition is that distinctions between certain pairs of tags, which cause the most inconsistencies in annotation, are not necessary. These subtle differentiations may be linguistically motivated at some points, but there are at least a couple of issues which come up in respect to making fine distinctions in tagset construction:

- Are certain distinctions in the tagset indeed essential? There is no straightforward answer to this question as tagsets are often dependent on a certain application. So it may make sense to keep tagsets as gross as possible and extend them when needed.
- If they are indeed necessary, then the annotators should be extra trained for the task, so that the problems (2) and (3) could disappear.

Concerning Problem (2), we cannot influence it unless we compile each time own corpora but even then it is difficult to eliminate errors. The latter implies that we should take it for granted that only up to ca. 98% of accuracy for tagging is possible. What's more, the true numbers for unseen texts and languages other than English seem to be even worse, i.e. 96% accuracy is actually good if it is guaranteed.

The question is, whether we obtain such high accuracy also in "real-life" application and not only under the lab conditions? This is what we are going to dwell on in detail in the next chapters.

- Even if we had completely consistent corpora and a 100% agreement between annotators, whether existing taggers still would be able to cope with certain distinctions which are often not possible without background *world* knowledge? Presumably they would, but under the same conditions as humans, i.e. the taggers would need to be trained extra for these special cases.

2.3 Summary

	overall	unknown (% of)	known
TNT (Brants, 2000)	96.70	85.50 (2.9%)	97.0
TreeTagger (Schmid, 1995)	96.81	–	–
TBL (Brill, 1995)	96.50	85.00	–
MaxEnt (Ratnaparkhi, 1996)	96.63	85.56 (2.65%)	96.86
SVM (Gimenez and Marquez, 2004)	97.16	89.01	97.39
StanfordTagger	97.24	89.04	–
ENGCG Tagger	98.54	–	–

Table 2.16: Reported Published Evaluations of POS Taggers for English (Penn Treebank)

The accuracies for English taggers range from 96.50% to 97.24% and thereby differ at most in 0.74% (except for the ENGCG Tagger the accuracy of which has never been reproduced). However, all of the tests have been conducted on the known for its inconsistency PENN TREEBANK. Furthermore, the used test set of the PENN TREEBANK has quite a small amount of unknown words.

	overall	unknown	known
TNT (Brants, 2000)	96.7	89.0 (11.9%)	97.7
Treetagger (Schmid, 1995)	97.53	78 (2%)	97.4
TBL in (Volk and Schneider, 1998)	94.57	81.52 (15%)	–
Treetagger in (Volk and Schneider, 1998)	95.27%	84.05 (15%)	–

Table 2.17: Reported Published Evaluations of POS Taggers for German

In case of German, the number of unknown words is consistently bigger (except for the first TREETAGGER experiment of Schmid (1995)) which could be one of the reasons for greater variation in overall accuracies.

There are two independent evaluations available for the TREETAGGER. It is interesting to see the range of the outcomes: 97.53% in the original paper vs. 95.27% in Volk and Schneider (1998) experiment. Some differences in the tuning parameters could count partially for that but surely not only. A further possible reason could be a bigger number of unknown words in Volk’s evaluation.

Transformation-based tagger is performing worse for German, with overall accuracy being 2% less than for English.

What is interesting, is that TNT tagger is performing consistently good on both English and German, with even significantly better accuracy for unknown words in German. The latter compensates for the larger percent of unknowns in German.

As to our knowledge, there have been no published results so far for any other tagging approach for German, except for HMM-based taggers and occasionally transformation-based ones.

Furthermore, most evaluations described in this chapter were performed on the same text collection on which the respective models had been trained; and often the separation into the training and testing sets has been done on the level of sentences, i.e. 9 of 10 sentences have been used for training and every tenth sentence has been employed for testing. This introduces a kind of text collection or genre bias in evaluation.

Chapter 3

Resources, Tools, Methodology

3.1 Experiment Setup

We take over the methodology for tagger evaluation specified in [EAGLES \(Version of May, 1996\)](#) which consists of three main directions:

1. **Tagger evaluation:** comparing different taggers on the same training and test datasets with the same tagset;
2. **Tagset evaluation:** testing the same text on the same corpus with modified tagsets;
3. **Text type evaluation:** using the same tagger with the same tagset to train it on one text type and test on different text types.

Since we are testing TIGER vs. DEWAC corpus tagging, we are dealing here with text type evaluation most of the time.

3.1.1 Tagger Evaluation

We evaluate in detail the TREETAGGER (s. Section [2.1.1.4](#)) with a Standard Parameter File (SPF) as well as by cross-validation with a number of settings. In cross validation experiments we find the tagging accuracy averaged over 10 runs. The overall accuracy as well as the accuracies for known and unknown words are calculated. Cross validation is performed on 10 contiguous sections of corpora

with each iteration using 90% for training and 10% for testing, thereby ensuring unseen text in the test phase.

Alternatively to the contiguous divisions, one could use a round-robin procedure that puts every 10th sentence into the test set. The latter would imply that all the texts would have been at least partially seen by the tagger before, thereby the tagger would have been "indirectly" trained on the more or less the same texts. Taking coherent parts of the text for training and testing represents thereby more realistic setup (Brants, 2000).

Accuracy is defined as the number of the correctly tagged tokens divided by the total number of tokens in the test corpus.

We test three different experiment setups in cross validation of the TreeTagger, using its feature of integrating external lexicon for training:

1. cross validation is performed using only training corpus;

This has always been a standard evaluation procedure. In this case, however, the resulting model is biased towards the training corpus.

2. external big German lexicon is integrated for training;

This allows us to measure the influence of big external dictionaries on tagging performance of the TreeTagger, and thereby figure out whether it makes sense to invest effort in constructing external dictionaries.

3. lexicon is extracted from the whole evaluation corpus and constitutes words which have been successfully analyzed by SMOR (Helmut Schmid and Heid, 2004) morphological analyzer.

The hope here is, on the one side, that there are less unknown words for the tagger (assumed that SMOR analyzes the most words successfully) and less unknown tags for certain ambiguous words (e.g. words that are present in the training corpus only with one tag, but in reality can have a couple of different ones). So, in this setting we try to reduce the number of unknowns and consequently the number of potential *lexical errors*. On the other side, we introduce much more ambiguity into the model. The latter implies, that we can thereby test the disambiguating capacity of the tagger under the worst conditions, i.e. when it has a lot of choices.

Consequently, we evaluate the influence of the pre-compiled dictionaries as well as morphology on the tagger’s performance by employing different lexicons.

Further, we test STANFORD TAGGER as an example of a tagger based on a completely different mathematical model (maximum entropy) and with the best reported accuracy for English till now. We are not aware of any publications on the performance of this tagger for German.

As a next step, we test Apache UIMA Tagger ([Apache](#)), an open source hidden Markov model tagger which is a state-of-the-art HMM tagger, inspired by TNT TAGGER([Brants, 2000](#)), that is not using any complex strategies except for the longest suffix compliance for unknown words and deleted interpolation for smoothing as well as a couple of simple heuristics.

Thereby, we achieve a comparison for the performance of the two best published state-of-the-art statistical taggers (Decision Tree HMM tagger and a Max-Ent tagger) as well as one open-source HMM tagger.

3.1.2 Tagset Evaluation

It is obvious from a number of evaluations that a lot of tagging errors are caused by sometimes too fine differentiations within major categories, as in many of these cases the local distribution is not distinctive any more and the state-of-the-art taggers use mostly only local context.

Thus, in the next step we want to evaluate the influence of the tagset granularity on the tagging accuracy.

It is usually agreed that a used tagset is at dependant on the language and purpose of tagging. One can differentiate, however, a number of ”universal” parts of speech, independent of the context and at least for West European languages. These are:

- Open word classes: noun, verb, adjective, interjection, adverb
- Closed word classes: determiner, pronoun, modal verb, auxiliary verb, preposition, particle, conjunction, numeral

A word class is closed if it is a fixed set, and no new words are introduced into this class. Most part of speech tagsets are refinements of these traditional classes,

for instance they can distinguish various types of pronouns, verbs, etc. A tagset can be very small (up to 20 tags) or very detailed (over 200 tags). The early tagsets made in general finer distinctions (e.g. certain qualifiers or determiners), more recent tagsets have made fewer distinctions (Manning and Schütze, 1999).

As an example, the pioneering BROWN CORPUS distinguishes 87 simple tags and allows compound tags, e.g. I'M would be tagged there as PPSS+BEM (PPSS for *non-third person nominative personal pronoun* and BEM for *am*), thereby resulting in a couple of hundreds of possible tags. The Lancaster-Oslo/Bergen(LOB) Corpus elaborated Brown Corpus further and is composed of 135 basic tags (without compound tags); the Lancaster UCREL includes 165 tags and the London-Lund Corpus of Spoken English comprises 197 different tags. The idea behind introducing so many fine distinctions into the tagset is "the ideal of providing distinct codings for all classes of words having distinct grammatical behaviour" (Garside and Sampson, 1987). However, the most widely used tagset for English at the moment is the PENN TREEBANK tagset. Like most other English tagsets, it was deduced from the BROWN CORPUS and contains only 36 part of speech tags and 12 other tags for punctuation and currency symbols.

The history of German tagsets is not as long as for English. The most famous and presumably the only one used for German is the 1995 version of STTS (Stuttgart-Tuebingen Tagset), which has resulted out of its predecessor, IMSTUE tagset of 1993 (also jointly developed by Stuttgart and Tuebingen). Current state-of-the-art German taggers or corpora use STTS, sometimes with slight modifications. STTS tagset was developed in accordance with the EAGLES Initiative.

EAGLES (Version of May, 1996) examined the questions of relevance for tagset design. They pointed out that a good tagset should find a balance between features useful for linguistic processing and the ones which the tagger can technically provide.

In short, these reasons can be summarized in the following three questions:

1. How much linguistics should be integrated in the tagset?
2. Which tags affect the overall performance of taggers negatively?

3. Which lexical ambiguities can be solved by distributional classification, i.e. which distinctions are feasible for a tagger due to their distinct distribution.

Therefore, possible directions in tagset modifications include:

- **Type I: Granularity Changes**

Ia *Simple*: tags for ambiguous words are merged or v.v. split into one or more unambiguous tags, the latter is advisable only if their distribution is distinct enough;

Ib *Complex*: overlapping tags as, for example, in case of personal(A) and reflexive(B) pronouns, can be modified in three ways: form one ambiguous AB, split into unambiguous A, B and ambiguous AB, or merge the ambiguous AB with either unambiguous A (into (A—AB)) or unambiguous B (into B—AB);

Ic *External*: it might be worth splitting an unambiguous class if it impacts positively disambiguation of another ambiguous class.

- **Type II: Changes in the assignment of word forms to tags**

As an example, it is difficult to distinguish in German lexicalized participles used adverbially (ADJD) from verbal participles (VVPP), so one could try to order word forms to specific tags differently depending on the syntactic constructions in which such ambiguous words appear.

As an example, they evaluate the quality of the following tag pairs:

- NN vs NE (*type Ia*) - these two tags have very similar distributions, which makes this distinction very hard for a POS tagger. Furthermore, there is no orthographic difference in German either. The intuition would be to postpone this step to further more elaborate processing. There is a bunch of research nowadays in the field of Named Entity Recognition.
- VA-, VM-, VV- verb forms - were conflated with the corresponding VV-forms.

The experiment with merging of noun classes led to the slight overall accuracy improvement and to the almost double increase in accuracy for the NOUN class which is a good indication for merging the categories. The setting with VERB forms didn't bring any improvement as different verb classes seem to be more distinctive in distribution, so it does not make a big sense to conflate them. Unfortunately, **EAGLES** (*Version of May, 1996*) haven't tested all of the conditions that they had outlined.

The only tagset reduction or fusion setting, we test in our experiments, is the reduction to basic part-of-speech classes.

Mapping onto the major grammatical categories is implemented: NOUN, VERB, ARTICLE, ADJECTIVE, PRONOUN, CARDINAL, ADVERB, CONJUNCTION, PREPOSITION, INTERJECTION, PARTICLE plus FM (foreign material), TRUNC, XY and one punctuation tag (\$) instead of three.

The above-defined mapping is just one out of many possible and is motivated by the idea that finer distinctions in the tagset are not necessary in the majority of "real life" applications.

3.1.3 Text Type Evaluation

Apart from doing text type evaluation on the level of a complete TIGER corpus against the DEWAC collection, we do evaluation of tagging performance especially for web texts of various nature. DEWAC corpus is a good opportunity to do it, as the web consists of enormous number of various genres. Furthermore, DEWAC keeps the text delimiters annotation, which makes this kind of evaluation possible at all, the latter is not the case in most corpora.

3.2 Corpora

In the following, we briefly describe corpora used for our experiments. As our primary goal is to evaluate tagging accuracy on web corpora, we need a manually annotated corpus of random web texts. To our knowledge, there is no such corpus readily available. Consequently, the first task was to manually annotate a cutout of such a web collection. We use a sample of DeWaC corpus (**Baroni and Kilgarriff**,

2006), which is a collection of texts, which has been crawled directly from the World Wide Web. The data have been cleaned up, namely HTML code as well as "boilerplates"¹ have been removed. Finally, language detection was done, as well as stripping of near-duplicates (i.e. identical web pages) and *spam* sites, such as pornography or pages with automatically generated content. In the next step, the collection was tagged with TREETAGGER (Schmid, 1995).

We extracted about 11000 tokens of contiguous text from the DeWaC corpus (the collection has been randomized before) and manually corrected and double checked automatic tagging word-by-word.

During the process of manual re-annotation, some further changes were undertaken on the text, namely:

- the maximal length of the text for a single web page was limited to 2000;
- the errors in tokenization were corrected;
- text had to be cleaned partially from the remaining boilerplates (according to the web page browser view where available).

We also measured the interannotator agreement on a small subsection of our web corpus (DEWAC). Note, that unlike in Marcus et al. (1994) the annotators have not been extra-trained on the task, but both have graduate training in linguistics. The agreement was about 96.7%.

Further, we decide to evaluate also a "standard" newspaper corpus, in order to get a direct comparison of the outcomes for a reference newspaper corpus and web corpus under the same conditions. The corpus of our choice is the TIGER TREEBANK (Brants et al., 2002). It consists of about 900,000 tokens (50,000 sentences) of German newspaper text, taken from the *Frankfurter Rundschau*. The corpus was annotated with syntactic structure and semi-automatically POS-tagged. Each sentence has been manually corrected by two annotators independently, followed by a consistency check (Brants and Hansen, 2002).

¹A "biolerplate" is a meaningless content that carries no semantic information and is mostly automatically generated, like navigation information, copyright, advertisements, etc

Though we have seen a number of reported accuracies for typical newspaper *gold standard* corpora in Chapter 2, we decide to perform our own experiments for a number of reasons:

1. the reported numbers often disagree because of different experiment settings such as different corpora or different parts of corpora, partially different algorithms, evaluation methodology (e.g. cross validation vs. one test set);
2. no proper evaluation has been done for the TIGER corpus yet, though Tiger Treebank is the most valuable resource for German linguistics at the moment (due to its size and coverage). Brants (2000) evaluated his TNT tagger on the NEGRA corpus though, which was the predecessor of TIGER but much smaller. The TREETAGGER was evaluated only on a small fragment of manually annotated *Stuttgarter Zeitung*.

Consequently, to get more reliable comparison, we employ both a "classical" newspaper corpus and a web corpus in our testing.

Chapter 4

Tiger Evaluation

In this chapter, a thorough evaluation of `TREETAGGER` on the `TIGER` corpus is done, which comprises tagging with the standard parameter file and 10-fold cross validation with different lexicons. We also do a rough tagset evaluation with `TREETAGGER`.

Further, we report performance for `STANFORD TAGGER` and `APACHE UIMA TAGGER` on the `TIGER`. At the end of the chapter, the results are compared and discussed.

4.1 Tagger Evaluation

4.1.1 TreeTagger

We first evaluate `TREETAGGER` on the `TIGER` corpus with the big German standard parameter file ¹.

Afterwards we evaluate by means of 10-fold cross validation. For every experimental run, overall accuracy and accuracies for known and unknown words are reported. For unknown words, the percentage of the latter in the test corpus is indicated. Furthermore, the percentage of lexical and disambiguation errors is

¹There are two German parameter files for Windows/Linux on the official website of the `TreeTagger`: <http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/>. We chose to use here the BIG German standard parameter file as it performed slightly better than the other one.

4.1 Tagger Evaluation

determined. We denote the errors which are caused by the words listed in the lexicon but not with the right tag as *lexical errors* (LE). Errors for unknown words are also cases of lexical errors, but we list them separately here, in order to see how many errors are caused by incomplete or imperfect dictionaries. *Disambiguation errors* (DE), in contrast, are the cases where the tagger fails to choose the right tag out of two or more possible.

In case of cross-validation, the corpus is divided into 10 contiguous parts. Then the cross validation is performed with three different setups as described in Section 3.1.1:

1. only the vocabulary extracted from the nine training parts is used as a lexicon (82000-85000 entries);
2. the general German lexicon from the original TREETAGGER evaluation (780407 entries) is used for training,
3. tokens from the whole TIGER corpus which were successfully analyzed by a morphological component constitute the training dictionary.

	overall	known	unknown	UT ^{a)}	LE ^{b)}	UE ^{c)}	DE ^{d)}
Schmid (1995)							
	97.53%	97.4%	78%	2%	–	–	–
Standard Parameter File (SPF)							
	95.82%	96.27%	79.88%	2.7%	1.38%	0.55%	2.25%
<i>10-fold Cross Validation</i>							
Setting 1	96.90%	97.62%	87.89%	7.44%	0.52%	0.90%	1.68%
Setting 2	95.85%	96.88%	69.35%	3.76%	1.72%	1.15%	1.28%
Setting 3	93.19%	96.50%	25.16%	4.63%	2.03%	3.47%	1.31%

Table 4.1: TreeTagger Evaluation on Tiger

^{a)}Unknown Tokens: percent of unknown tokens in respect to all tokens in the corpus

^{b)}Lexical Errors: here just the words listed in the lexicon but not with the right tag

^{c)}Unknown Errors: percent of errors for unknown words in respect to all tokens in the corpus

^{d)}Disambiguation Errors: proper errors of the tagging algorithm

Table 4.1 demonstrates the range of the resulting accuracies between 93 and 97 percent¹. No one of our experiment setups achieves a comparable accuracy to the one reported in Schmid (1995), except for the accuracy for known words in CV Setting 1. It is also worth mentioning that the accuracy for unknown words is the highest in this condition, but the much bigger proportion of the latter makes the overall frequency reside under the reported one. This evaluation setup corresponds at most to the original one, as the same corpus without external noise is used both for training and tagging, which also explains the relatively high accuracy for unknown words, compared to the other experiments.

Tagging with the standard parameter file results in the total accuracy of **95.82%** which is 1.71% less than the published one; the accuracy for unknown words is in contrast slightly better. The similar overall outcome as with standard parameter file is observed in cross validation experiment where the general German lexicon is employed as an external lexicon resource in the process of training on the TIGER corpus. This is not surprising as more or less the same lexicon was used for the construction of the original standard parameter file of the TREE-TAGGER. Though the average accuracy in these cases is almost the same, the distribution of lexical and disambiguation errors is different. The number of errors for unknown words is significantly less for the standard parameter file, which could be explained by the fact that there are less unknown words. In contrast, the number of disambiguation errors is almost one percent more, which presumably indicates that training on the same type of corpus improves disambiguation abilities of the tagger as it learns specific for this text type part-of-speech sequences.

Consequently, the best accuracy was achieved when doing cross validation with the corpus' own lexicon. Though the percentage of unknown words under this condition is the biggest, the accuracy for them is relatively high so that the error for unknown words remains very small. Lexical error rate is very small in this condition, as the tagger learns only a certain range of possible parts of speech for certain lexicon entries which reduces noise in the vocabulary.

The worst observed result is for the cross validation under condition 3, i.e. where we use an external lexicon compiled with the help of morphological analysis. In this case, disambiguation error remains very little but the number of lexical

¹Please note, that all the numbers in this and following tables are rounded values.

and unknown word errors increases dramatically. The crucially bad performance of the tagger on unknown words here is especially striking. Thus, morphology seems to introduce a lot of noise and degrade the tagger’s accuracy.

4.1.1.1 Disambiguation Error Statistics for SPF

Further, the outcome of the tagger for the big standard parameter file is analyzed in detail, as the situation where we have just a standard parameter file at hand is more realistic.

Tables 4.2 and 4.3 present an overview of the disambiguation error statistics. In Table 4.2, the ambiguity rate error statistics is shown, i.e. how many errors occur in each ambiguity class, where an ambiguity class means a number of available tags per word.

ambiguity	number of errors	in % (in respect to all tokens)
1	67	0.007
2	7534	0.83
3	8137	0.91
4	3689	0.40
5	596	0.06

Table 4.2: Disambiguation Error Statistics for `TREETAGGER` and `TIGER` Corpus with Standard Parameter File (using after-processing)

As it is evident from table 4.3, that the tagger has the most problems with the unknown words and the words of the ambiguity class "4", which make up together about 5 percent.

What is confusing here is that there are 67 errors among the tokens of the ambiguity class "1", which is in principle impossible, as there is no ambiguity if only one tag is available. If only a wrong tag is available to the tagger, then it should be a lexical error. After the manual inspection of these errors, it turned out that all of them are due to the after-processing script for German included into the tagger package:

4.1 Tagger Evaluation

ambig. class	tokens (% of all)	correct	in %	errors	in %
0	24086 (2.7)	19241	79.88	4845	20.12
1	538180 (60.5)	538113	99.99	67	0.01
2	163788 (18.5)	156254	95.5	7534	4.5
3	133519 (15)	125382	94	8137	6
4	19148 (2.2)	15459	80.8	3689	19.2
5	9857 (1.1)	9261	94	596	6
total	888578 (100.0)	851476	95.82	37102	4.18

Table 4.3: Ambiguity Classes Statistics

```

697489 schossen ['VVFIN'] VVFIN VVINF
723650 wurden ['VAFIN'] VAFIN VAINF
728980 lieen ['VVFIN'] VVFIN VVINF
735198 wurden ['VAFIN'] VAFIN VAINF
744879 schlugen ['VVFIN'] VVFIN VVINF
790471 waren ['VAFIN'] VAFIN VAINF
801593 abstieen ['VVFIN'] VVFIN VVINF
811271 zurckflossen ['VVFIN'] VVFIN VVINF

```

It obviously happens from time to time, that this script modifies the right tags into the wrong tags which are not available for that token at all (the *available* tag is given in parenthesis, the second column indicates the *gold standard* tag and the last column shows the *tagger tag*).

On the other side, if one completely deactivates this after-processing the results are slightly worse (with *accuracy* = 95.63%, which is 0.2% less). The fact that these errors disappear if we do not use after-processing proves that they are indeed caused by this script. As table 4.4 shows, the ambiguity class "1" completely disappears without using the script, but at the same time there are more disambiguation errors in other classes, especially in ambiguity class "2" (see table 4.4).

Thus, we further analyze the tagger's output with the standard parameter file and using after-processing script.

ambiguity	number of errors	in %
2	9209	1
3	8212	0.9
4	3691	0.4
5	596	0.06

Table 4.4: Disambiguation Error Statistics for TREE_TAGGER and TIGER Corpus with Standard Parameter File (without after - processing)

4.1.1.2 Error Analysis

After having collected error statistics for individual tags, we detected the similar error sources as in Schmid (1995) and Volk and Schneider (1998). As tables 4.5 and 4.6 indicate, most of the errors are due to the non-sufficient distributional differences within categories. The major sources of errors for TIGER are:

1. mixing up common and proper nouns;
2. interchanging different verb forms;
3. confusing adverbs with adverbially used adjectives;
4. the confusion between APPR and KOKOM ended up quite unexpected at the second place with more than 2000 errors! After closer examination of this error, it turned out that the annotation of e.g. *als* is very noteworthy in TIGER. Thus, in the following cases, instead of being annotated as KOKOM as one would normally expect, it has been annotated as a preposition APPR (s. figure 4.1).

There are a lot of such cases in the corpus. Obviously, the annotators of the TIGER corpus introduced a new finer distinction in the annotation of *als*, which caused more problems to the tagger.

```

<s id="50419">
Hinter APPR hinter
Scharping NE Scharping
gelten VVFIN gelten
nur ADV nur
... ..
und KON und
Wolfgang NE Wolfgang
Thierse NE Thierse
als APPR als
ungefhrdet ADJD ungefhrdet
. \$. --
</s>

or

<s id="2">
Konzernchefs NN Konzernchef
lehnen VVFIN lehnen
den ART der
Milliandr NN Milliandr
als APPR als
US-Prsidenten NN US-Prsident
ab PTKVZ ab
</s>

```

Figure 4.1: Examples of *als* Annotation in TIGER

tag	number
NE	8008
APPR	3574
VVFIN	2509
ADV	2445
NN	1925
ADJD	1787
ADJA	1665
PIS	1302
VVINFIN	1217
VVPP	1201

Table 4.5: Tags with the most errors (TIGER, Standard Parameter File)

number	correct tag	tagger tag
6605	NE	NN
2517	APPR	KOKOM
1223	NN	NE
1015	VVINFIN	VVFIN
967	VVFIN	VVPP
912	ADJA	NN
910	PWAV	KOUS
863	ADV	ADJD
854	ADJD	ADV
786	VVFIN	VVINFIN

Table 4.6: Most frequent error types

number	word	correct tag	tagger tag
2236	als	APPR	KOKOM
615	wie	PWAV	KOUS
567	rund	ADJD	ADV
430	bis	APPR	KON
295	Wie	PWAV	KOUS
284	aber	KON	ADV
281	Als	APPR	KOKOM
280	mehr	PIAT	ADV
251	die	ART	PRELS
233	mehr	PIS	ADV
232	die	PRELS	ART
164	werden	VAINF	VAFIN

Table 4.7: Mostly misclassified words

4.1.2 Stanford MaxEnt Tagger

There is, to our knowledge, no published results for using MaxEnt taggers for German. That is why it was especially interesting to test our German corpora with a variant of a Maximum Entropy tagger, which is moreover the tagger with the best reported accuracies at the moment (s. 2.1.3). As no standard parameter file for German is provided for STANFORD TAGGER, we cannot evaluate a standard parameter file setting as with TREETAGGER. The only possible evaluation for TIGER here is to train and test on the same corpus.

The procedure is the following:

1. To train the tagger, set up a training file in the format one sentence per line with words separated by spaces and each tag suffixed to the word by an underscore.

E.g.: I_PRP wish_VBP I_PRP were_VBD programming_VBG in_IN Ruby_NNP
...

2. Modify TTags.java for German closed classes (as there are only english, polish and chinese at the moment in the package).

3. Train and test as described in the manual.

Training of the tagger has taken about 5-6 hours on, then it was first tested on exactly the same corpus and achieved accuracy of **99%**, which is about 0.8 percent better than the `TREETAGGER` under the same conditions (**98.2%**). The latter shows that `STANFORD TAGGER` is a bit better at disambiguation, which could be due to the fact that a bidirectional model of it was used, in contrast to `TREETAGGER` that is only moving in one direction.

As a next evaluation step, we perform *10-fold cross validation* (see table 4.8 for the results).

	% of unknown words	unknown accuracy	overall accuracy
random sentences	6	93%	97,89%
text blocks	7.52	91.66%	97.625%

Table 4.8: 10-fold Cross Validation for TIGER with Stanford Tagger

It should be noted, however, that training and testing with MaxEnt tagger takes much more time than with a HMM/Decision Tree tagger, (i.e. hours vs seconds). That makes MaxEnt unusable for online applications (at least in its current form).

4.1.3 Tagger Evaluation Results

To summarize, we compared three taggers under the same condition, namely trained on the same corpus and using only the lexicon from the training parts, i.e. the test corpus always contained unseen material:

1. TreeTagger
2. StanfordTagger
3. Apache UIMA HMM Tagger

Table 4.9 contains mean overall accuracy for the 10-fold cross validation as well as the mean accuracies for known and unknown words and the percent of

4.1 Tagger Evaluation

	TreeTagger	StanfordTagger	Apache HMM Tagger
unknown words (σ)	7.44 (0.78)	7.52 (0.46)	7.66 (1.38)
unknown accuracy (σ)	87.89% (0.99)	91.664% (0.83)	95.18 (0.39)
known accuracy (σ)	97.62% (0.21)	–	96.10% (0.38)
total accuracy (σ)	96.89% (0.34)	97.625% (0.24)	96.03% (0.38)

Table 4.9: Tagger Evaluation on TIGER

unknown words for each of the taggers. The standard deviation is also given for every resulting parameter.

Obviously, the best overall result is achieved with the bidirectional MaxEnt STANFORD TAGGER whose average total accuracy is as high as the one for the known words with TREETAGGER. Remarkable is the accuracy of the APACHE UIMA HMM TAGGER for unknown words which is almost as high as the accuracy for known words. It is also worth mentioning that the disambiguation error of the UIMA Tagger was under one percent in cross validation experiments. Apparently, the low performance of the TREETAGGER on unknown words is *the* problem of this tagger when compared to other statistical taggers. In contrast, the TREETAGGER was reported to be better at tagging unknown words when compared to the rule-based BRILL TAGGER (Volk and Schneider, 1998).

4.2 Tagset Evaluation

In the next step, we want to evaluate the influence of the tagset granularity on the tagger’s accuracy. Mapping onto the major grammatical categories is implemented, as described in section 3.1.2.

There are three experiment setups which we test:

1. After tagging with the original STTS tagset (54 tags) and the standard parameter file, we map the *output* of the tagger onto a reduced tagset. As a result, the accuracy is getting about two percent better, and for unknown words even almost 14% better (see table 4.10 vs. table 4.1).
2. Mapping is done before training, so that the tagger learns a model for a reduced tagset; then it is tested with cross validation and its own lexicon (i.e. the lexicon formed from the training parts of the corpus).
3. Cross validation is performed with own lexicon (as in 2); resulting tags are mapped after tagging. The outcome is more or less the upper bound of the TREE_TAGGER, as it achieves the same accuracy when tested on the same corpus as trained, i.e. without unknown words at all.

	overall	unknown (% of unknowns)	known	LE	DE
Mapping 1	97.79	93.50 (2.7)	97.8	1.0	1.2
Mapping 2	97.34	94.92 (7.4)	97.5	0.84	1.82
Mapping 3	98.28	95.60 (7.4)	98.5	0.36	1.35

Table 4.10: TREE_TAGGER Accuracy for a Mapped Tagset

Obviously, a better output is achieved when training the tagger on a finer tagset and mapping afterwards.

Apart from accuracy, a kappa coefficient (Cohen, 1960) was computed for tagging results with a mapped tagset for Setting 1. As certain annotations could coincide just by chance, a kappa coefficient is often used to measure a "true" agreement. Kappa takes into consideration not just the observed outcome but

4.2 Tagset Evaluation

also the expected one. The idea behind was that kappa might correct for the easier tagging task on the simplified tagset. For the standard tagset, difference between observed accuracy and kappa is minimal (observed accuracy is 0.958 and kappa is 0.95). However, kappa for a mapped tagset was about 0.97, which shows again the same minimal difference between observed accuracy and kappa.

	Adjective	Adverb	CARD	Noun	Preposition	Pronoun	Verb	\$	Conjunction	XY	Article	FM	Interjection	TRUNC	total
Adjective	70844	957	20	811	33	372	643		6	50	7	79	1	7	73830
Adverb	866	36050		38	120	1429	2		715	2		13	2	1	39238
CARD	6		15837	15		10								1	15869
Noun	1098	74	38	234069	42	58	82	3	13	544	8	931	5	29	236994
Preposition	44	541		129	87396	593	1		246	5		11			88966
Pronoun	218	498	3	67	291	68158	35		60	2	659	10		1	70002
Verb	880	27		181	3	68	106170		4	17		29	2		107381
\$					3	1		119602		3					119609
Conjunction		341		28	3229	1046			31703	9		4			36360
XY	1	4	3	66		4				233	1	48			360
Article	8	2	30	56		882					96622	2			97602
FM			8	694	4	1				146	1	136	1		991
Interjection		1		7						2		2	7		19
TRUNC	8			13										1325	1346
total	73973	38495	15939	236174	91121	72622	106933	119605	32747	1013	97298	1265	18	1364	888567

Figure 4.2: Confusion Matrix for Tiger Tagging with a Mapped Tagset

4.3 Summary

We thoroughly evaluated tagging performance of TreeTagger for the TIGER corpus in this section by both tagging it with standard parameter file and 10-fold cross validation. The outcomes are summarized in the following:

- when tagging with standard parameter file, the accuracy is 95.82% which is 1.71% less than the published one;
- having a big external dictionary of good quality as an addition to the training corpus is helpful enough to get similar results as with the standard parameter file (95.85% accuracy vs. 95.82% with SPF);
- using of morphology for constructing such a dictionary turns out to decrease substantially the precision of tagging compared to just using internal vocabulary;
- as most of the errors are due to the non-sufficient distributional differences within categories, the hypothesis is to reduce the tagset to just necessary gross tags containing the main grammatical categories. Such a tagset mapping indeed decreases the number of tagging errors significantly;

Furthermore, we compared tagging accuracy of TREETAGGER with that of STANFORD TAGGER as well as APACHE UIMA HMM TAGGER. The evaluation with these taggers has never been done before for German. The difference in the resulting accuracies for all taggers was in the range of two percent (96.03% for UIMA HMM TAGGER, 96.89% for TREETAGGER and 97.62% for STANFORD TAGGER). The best overall outcome was achieved thereby with the STANFORD TAGGER. However, the hours of training and tagging required by the current implementation of this MaxEnt tagger make it non-applicable anyway for real-time applications, and the gain in accuracy of ca. 0.7% is presumably not worth it. Interestingly, the best accuracy for unknown words was shown by the UIMA TAGGER.

Chapter 5

DeWaC Evaluation

In this chapter, we evaluate DEWAC (the German web corpus), described in Chapter 3.2. The procedure is the same as with the TIGER corpus, i.e. the TreeTagger is tested with the standard parameter file first, then 10-fold cross validation is performed with different lexicons. We also do tagset evaluation with TREETAGGER. Last but not least, we estimate the tagger’s performance for different text types or genres. In the end, the results are compared and discussed.

5.1 Tagger Evaluation

5.1.1 TreeTagger

We proceed in the same way as the experiments with TIGER (see Chapter 4). First the tagger’s behaviour on the DeWaC corpus is measured with the standard parameter file; afterwards 10-fold cross validation is done. For every experimental run, overall accuracy and accuracies for known and unknown words are reported as well as the proportion of lexical and disambiguation errors. For unknown words, we report how many of them are in the test corpus (in percent).

In case of cross-validation, the corpus is divided into 10 contiguous parts. Then the experiment is run with the following settings (see also Section 3.1.1 on Tagger Evaluation strategy):

5.1 Tagger Evaluation

1. only the vocabulary extracted from the nine training parts is used as a lexicon (ca. 3000-3200 entries per each iteration);
2. the general German lexicon containing 780407 entries is used for training;
3. tokens from the whole DEWAC corpus which were successfully analyzed by a morphological component constitute the lexicon;
4. tokens from the TIGER corpus which were successfully analyzed by a morphological component make up the lexicon.

	test set	overall	known	unknown	UT ^{a)}	LE ^{b)}	UE ^{c)}	DE ^{d)}
Schmid (1995)	5000	97.53%	97.4%	78%	2			
<i>Tiger Corpus</i>								
SPF	888578	95.82%	96.27%	79.88%	2.7	1.38%	0.55%	2.25%
CV 1	888578	96.90%	97.62%	87.89%	7.44%	0.52%	0.90%	1.68%
<i>DeWaC Corpus</i>								
SPF	10057	93.71%	95.42%	54.3%	4.15%	2.63%	1.89%	1.76%
CV 1	10057	85.77%	93.00%	65.79%	28.70%	9.37%	0.76%	4.16%
CV 2	10057	92.16%	94.28%	54.39%	5.3%	2.89%	2.43%	2.51%
CV 3	10057	88.09%	94.17%	20.31%	8.22%	2.95%	6.55%	2.39%
CV 4	10057	88.54%	94.83%	54.29%	15.51%	1.93%	7.09%	2.43%

Table 5.1: Tagging accuracies for TREE_TAGGER on a section of DEWAC vs. TIGER

^{a)}Unknown Tokens: percent of unknown tokens in respect to all tokens in the corpus

^{b)}Lexical Errors: here just the words listed in the lexicon but not with the right tag

^{c)}Unknown Errors: percent of erroneously classified unknown words in respect to all tokens in the corpus

^{d)}Disambiguation Errors: proper errors of the tagging algorithm

The conclusions from the tagging results summed up in Table 5.1 are the following:

1. **Tiger vs DeWaC tagging:**

- when *tagging with standard parameter file*, the difference in disambiguation error of the tagger stays within the limit of at most 0.5 percent, but the overall accuracy is two percent less, which is due to lexical errors. The number of unknown words in the web corpus is surely larger, but the main hurdle is a very low accuracy for them;
- in contrast, *cross validation* results with the text's own lexicon from training parts are much worse (about 10 percent worse), mostly due to the huge number of lexical errors and also due to four times as much unknown words in the web corpus and the low accuracy for those. In both cases the accuracy for unknown words is 10% better with cross validation than with the standard parameter file.

2. DeWaC TreeTagger Evaluation:

- obviously here the best accuracy is achieved with the standard parameter file, the second-best result, just ca. one and a half percent less, is achieved with cross validation with the big external german lexicon. Contrary to expectations, really bad outcome is achieved with the text's own lexicon. A slightly better performance was observed when the training lexicon was extended by morphology, but still not good enough to exceed the standard parameter file. These findings suggest that in case where very few vocabulary is available for training, e.g. in our experiment it was a bit more than 3000 words, then using morphological analysis could held to improve the results slightly. If a more complete external dictionary is available, it definitely makes sense to use it for training. Last but not least, a good external parameter file trained on a different type of collection, seems still be sufficient for relatively good performance (93.71%).

5.1.2 Disambiguation Error Statistics

Proceeding in the same way as with the TIGER corpus, we compute the error statistics for the standard parameter file.

Tables 5.2 and 5.3 present an overview of the disambiguation error statistics.

In Table 5.2, the ambiguity rate error statistics is shown, i.e. how many errors occur in each ambiguity class, where an ambiguity class means a number of available tags per word and the ratio of such grammatically ambiguous words in the corpus. Table 5.3 summarizes the error statistics for different ambiguity classes.

ambiguity	number of errors	in % (in respect to all tokens)
1	10	0.10
2	86	0.85
3	54	0.53
4	19	0.19
5	8	0.08

Table 5.2: Disambiguation Error Statistics for TREE TAGGER and DEWAC Corpus with Standard Parameter File (using after-processing)

As it is evident from table 5.3, that the tagger has the most problems with unknown words and words of the ambiguity class "4" which were also most problematic for the TIGER corpus. The crucial impact to bad performance of the tagger on this corpus is due to the unknown words. Almost half of them are being tagged false. The error rate for 2-way and 3-way ambiguous words, which constitute the backbone of the corpus together with unambiguous words, is between 4 and 6 percent for both TIGER and DEWAC.

We further analyze the tagger's output with the standard parameter file in the next session.

5.1.3 Error Analysis

Table 5.4 lists top 10 of the tags with the most errors. These are almost the same as in case of TIGER but there are three new tags here (\$, FM and CARD) and two of them are even among top 3 tags with the most errors(c.f. Table 4.5).

Those are web corpus specific errors. \$ stands for punctuation, FM for foreign material and CARD for cardinals. All of them are prevalent in web, and obviously contribute to the lower tagging accuracy for web.

5.1 Tagger Evaluation

ambig. class	tokens	in %	correct	in %	errors	in %
0	418	4.15	226	54.30	191	45.70
1	6035	59.95	6025	99.83	10	0.16
2	1570	15.59	1484	94.52	86	5.48
3	1324	13.15	1288	95.92	54	4.08
4	158	1.57	139	87.97	19	12.03
5	118	1.17	110	93.22	8	6.78
total	10067	100.0	9434	93.71	633	6.29

Table 5.3: Ambiguity Classes Statistics

These three tags also appear in typical tag pair mistakes. Thus, if we compare the common tag pair confusions for TIGER (Table 4.6) and for DEWAC (Table 5.5), we find out that besides the usual disagreements already mentioned by Schmid (1995) and Volk and Schneider (1998) and confirmed by our TIGER experiments (i.e. *mixing up common and proper nouns* - still number one, interchanging different verb forms and adverbs with the adverbial use of adjectives), there are a lot of new errors due to the confusion of punctuation signs, foreign words and cardinals with common and proper nouns as well as adjectives.

tag	number
\$(132
NE	81
FM	74
NN	56
VVFIN	36
ADV	33
XY	29
CARD	25
ADJA	17
APPR	15

Table 5.4: Tags with the most errors (DEWAC, Standard Parameter File)

number	correct tag	tagger tag
73	NE	NN
70	\$()	\$.
41	FM	NN
29	NN	NE
28	FM	NE
24	CARD	NN
14	\$()	ADJA
13	ADV	ADJD
13	XY	NE
12	VVFIN	VVPP

Table 5.5: Most frequent error types in DEWAC

number of missclassifications	word
64	:
24	Eins
22	”
18	-
9	Re
8	—
7	Euro-sign
7	Krone
6	„
5	als
5	zu
5	Als

Table 5.6: Most frequently misclassified words from DEWAC

5.1.4 Tagger Comparison

For tagger evaluation with DEWAC we use again the aforementioned TREETAGGER, STANFORDTAGGER and APACHE UIMA TAGGER.

In order to reproduce a setup where we just have an external standard parameter file for our web corpus, we train all the three taggers on TIGER CORPUS and test on DEWAC.

In the next step, we perform a 10-fold cross validation using nine fragments of the corpus without any external dictionaries for training and the unseen one for test. For the comparison of the taggers' outcomes see Table 5.7.

Model trained on Tiger, tested on DeWac			
	overall	unknown	% of unknown words
TreeTagger	90.78%	69.12	11.48
Stanford Tagger	92.57%	75.34%	13.00
Apache UIMA HMM Tagger	91.6%	89.13%	13.43
Cross Validation			
	overall (σ)	unknown (σ)	% of unknown words
TreeTagger	85.77% (2.10)	65.79% (10.72)	28.70 (5.87)
Stanford Tagger	88.99% (2.26)	72.55% (9.53)	32.03 (4.89)
Apache UIMA HMM Tagger	85.05% (1.634)	83.98% (3.71)	32.03 (4.90)

Table 5.7: Tagger Comparison for DEWAC

For all taggers, models trained on TIGER perform significantly better for DEWAC than the cross validation experiments. The latter makes us think that the size and probably the quality of the training corpus makes a big difference. At a closer look on unknown words statistics, one could also infer that the number of unknowns also influences a lot the accuracy. As to the performance of the individual taggers, the winner in terms of the overall accuracies is in both cases the Stanford bidirectional tagger. The best accuracy for unknowns is achieved interestingly with the Apache UIMA HMM Tagger, which obviously has more problems with known words than the other two taggers, as the overall performance of the Apache UIMA Tagger in spite of a high unknown words' precision is still on the same level as the others.

5.2 Text Level Evaluation

As our section of DeWaC corpus contains almost all possible text types, it lends itself to conduct text type evaluation on this corpus (s. also section 3.1.3).

Table 5.8 shows the individual accuracies of each of the texts from our web corpus, tagged by TREE TAGGER.

text number	genre	overall accuracy	unknowns ^{a)}
1	movie description	93.89%	52.63% (36)
2	scientific news/medicine	96.88%	85.71% (2)
3	political speech(labor union)	97.52%	58.33% (5)
4	job market news	97.46%	80% (2)
5	story about Holy Paul	95.42%	68.62% (16)
6	biological exposition	94.23%	73.91% (6)
7	Rolling Stones tour (forum)	88.01%	39.2% (76)
8	child infections (report)	98.25%	– (0)
9	information about a conference	90.98%	33.33% (20)
10	IT news/Cebit	93.69%	46.42% (15)
11	news report (school district)	97.10%	33.33% (4)
12	news report (Archbishop)	91.97%	92.85% (1)
13	history (Gold War) report	96.67%	27.27% (8)

Table 5.8: DEWAC Text Level Accuracies (Total Accuracy and Accuracy for Unknown Words) with Standard Parameter File

^{a)}Statistics for unknown words: accuracy and the number of unknown words in the given text in paranthesis

The intuition behind performing this kind of evaluation is that the lower tagging performance on web is due to the heterogeneity of web in the sense of integrating different text genres. Text level statistics shows that the accuracy of the tagger on some kinds of texts is indeed very good, up to 98.25% which is the upper bound for this tagger. Traditional "newspaper" genres like reports, news, political speech, stories were tagged with high accuracy of 97% on average. What is important to notice is that these texts did not contain a lot of unknown words.

The good outcome for these genres may be due to the fact that the model for standard parameter file was trained on the same types of texts. The problem for taggers is obviously everything which is specific for the web (e.g. conference announcements, unstructured movie descriptions), or the so-called typical web 2.0 contributions, like forums, wikis and so on that often contain many errors.

The highest concentration of errors was in the forum text written all in lower case and by a non-native speaker:

e.g.

```
hallo/ITJ meine/PPOSAT name/NN ist/VAFIN nesko/ADJD ,/$,  
wohne/VVFIN in/APPR dubrovnik/NN in/APPR kroatien/NN ./$.  
habe/VAFIN schon/ADV stones/ADJA karte/NN fur/XY olympia/ADJD  
stadion/ADJA konzert/NN und/KON MOCHTE/VVFIN gerne/ADV auch/ADV  
fur/XY halle/VVFIN ...
```

In case of forums, wikis, or the like, it is especially important to have more flexible tagging models which are resistant to erroneous input. If the purpose of web corpus is to be used for linguistic research, one could just avoid crawling these texts, unless the purpose of the study is conversational text. If one is interested in information extraction, then it is necessary to cover web 2.0 content.

5.3 Tagset Evaluation

The experiments with TIGER prove that training models with a traditional more finegrained tagset and mapping after tagging attains the best results (the upper bound for the TREETAGGER). Taking into consideration this fact and that only a standard parameter file is available in most cases, we do a tagset evaluation for DEWAC for the following setup. After tagging with the original STTS tagset (54 tags) and the standard parameter file, we map the output of the tagger onto a reduced tagset. The general accuracy increase is about 2.8%, for unknown words - ca. 14%. Also note, that disambiguation errors are decreased (see Table 5.9). We also compute kappa coefficient for the mapped tagset. Kappa for the standard STTS tagset was 0.93 (observed accuracy = 93.7%); for a simplified tagset, kappa is 0.96 and the observed accuracy is 96.5%.

5.3 Tagset Evaluation

test tokens	overall	unknown (%)	known	LE	DE
10067	96.51%	66.50% (4.15)	97.81%	2.92	0.56

Table 5.9: Standard Parameter File Accuracy for DEWAC with Mapped Tagset

Importantly, the web specific most frequent errors for punctuation, foreign words and cardinals either disappear completely (i.e. for \$) or are significantly reduced, as for FM and CARD tags (c.f. Figure 5.1).

	Adjective	Adverb	CARD	Noun	Preposition	Pronoun	Verb	\$	Conjunction	XY	Article	FM	Interjection	TRUNC	Total
Adjective	676	14		20	1	3	5	23		5		3			750
Adverb	5	418		1	1	8	2		2			1			438
CARD			397	2											399
Noun	9	2	24	2555		2	4	17	1	20		69	2		2705
Preposition		1		1	955	4			4						965
Pronoun	4	12		1	6	852	1		1		3				880
Verb	4	2		10		1	1093	12	1	4					1127
\$								1354							1354
Conjunction		2			4	4		4	414						428
XY					2			8	1	0					11
Article						5			1		959	1			966
FM			1									19			20
Interjection													13		13
TRUNC														11	11
total	698	451	422	2590	969	879	1105	1418	425	29	962	93	15	11	10067

Figure 5.1: Confusion Matrix for DeWaC Tagging with a Mapped Tagset

Further, we gathered statistics for tagging performance with the simplified tagset for every single text from Table 5.8 under the same conditions as described above. The gain in accuracy is from ca. 1% (for the "traditional" texts) up to 6% for particularly difficult texts. In general, the lowest precision for a mapped tagset is 93.75% (c.f. 88.01% without mapping for the same *forum* text).

5.3 Tagset Evaluation

text number	without mapping		mapped after tagging	
	overall (in %)	unknown (in %)	overall	unknown
1	93.89	52.63	96.16	64.47
2	96.88	85.71	99.04	92.85
3	97.52	58.33	98.21	58.33
4	97.46	80	97.97	80
5	95.42	68.62	97.28	72.55
6	94.23	73.91	97.90	95.65
7	88.01	39.2	93.75	57.6
8	98.25	100	99.42	100
9	90.98	33.33	94.33	43.33
10	93.69	46.42	95.79	57.14
11	97.10	33.33	99.50	100
12	91.97	92.85	97.19	92.85
13	96.67	27.27	97.02	36.26

Table 5.10: Standard Parameter File Accuracies for a Mapped Tagset on the Level of Individual Texts

5.4 Summary

In this section, we evaluated the tagging performance on a section of web corpus. The lessons learnt include:

- when tagging web with standard parameter file, the accuracy is ca. 93.7% which is about two percent less than for TIGER, and 4% less than the reported one respectively;
- the big number of unknown words contributes considerably to the low accuracy; the accuracy for known words is more or less stable. Thus, a better treatment of unknowns, as well as symbols prevalent on web (i.e. specific punctuation, foreign material, cardinals, different encodings) is crucial;
- just having a big enough external dictionary as an addition to the small training corpus is very helpful (92.16% accuracy vs. 93.71% with SPF);
- using of morphology for constructing such a dictionary is only helpful if no "high quality" dictionary is available and if training corpus is very small (here: about 9000 tokens). As a contrast, in case of TIGER, using morphology reduced the precision greatly compared to just using internal vocabulary;
- gross tagset mapping decreases the number of tagging errors significantly;
- the true problem for web corpus is represented by uncontrolled content of web 2.0, i.e. forums, wikis, etc where a special treatment of punctuation, spelling, missing capitalization and ungrammaticality in the sense of untypical word grams is needed for robust tagging. Most texts or genres in a web corpus can be tagged rather well, while some others are disastrous. One could remove such texts from the corpus if they are irrelevant for particular purposes, or focus on recognising them and activating special heuristics, or different tagging models for those cases.

Chapter 6

Conclusions

The comparison of the published state-of-the-art part of speech taggers shows that, independent of the used approaches, tagging accuracies range between 95% and ca. 98% since 1988, and are not getting better.

Briefly, the reason is that all of them use more or less the same *features* that model only local context, except for the approach of [Tapanainen and Voutilainen \(1994\)](#), who are trying to combine both local (n-gram based) and global (grammar or rule-based) features and achieve pretty good results. Unfortunately, there is only one paper in this direction and neither open source software nor evaluation done by other authors are available for this tagger. However, even this tagger does not reach up to 100% accuracy.

The error sources which hinder the perfect performance of taggers can roughly be summarized in the following:

1. words that cannot be tagged correctly using only local dependencies;

These can be further subdivided into *intra-* and *inter-*category errors:

- a *intra* class errors are those which are due to the fine distinctions within the classes. We evaluated the influence of those by measuring tagger's performance for a reduced tagset that contains only major grammatical categories. It improves the accuracy significantly;
- b *inter* tag categories errors evidently constitute the rest ca. 2% of faults;

-
- (a) *heuristics errors*: there are quite a number of errors for the words which are in the dictionary and even with the only and the right tag, but are still being misclassified by the tagger:

e.g.

- i. Interestingly, the noun "Fisch" was tagged as VVIMP at the beginning of the sentence! Cases like these are examples of *heuristics* errors. Though the right tag is in the lexicon and actually it is the only tag available, the tagger still makes a mistake there, as the algorithm is presumably implemented in such a manner, that for the first words in the sentences it also looks up possible tags of the non-capitalized counterparts, which is obviously not always a good strategy.

```
<s id="49693">
Fisch  VVIMP  fischen
ist    VAFIN  sein noch
ADV    noch
frisch ADJD   frisch
,      $,    ,
wenn  KOUS  wenn
die   ART   d
Augen NN    Auge
des   ART   d
Fisches NN  Fisch
klar  ADJD   klar
...
```

If we force the tagger to output all possible tags for the first line of the above example, we get something like this:

```
Fisch: NN Fisch 0.801198 VVIMP fischen 0.198802
```

or the following cases:

```
<s id="5102">
IM  NE IM 1.000000
BLICKPUNKT NN Blickpunkt
1.000000 </s>
```

Such errors occur when heuristic rules are triggered by a tagger in certain situations, and simply fail for particular instances.

-
- ii. "tokenization" errors, including multiword expressions that lead to tagging errors:

e.g.

in DeWaC -

ziehen. NE ziehen.

or

etc). NE etc).

in Tiger -

Gold Standard

267033 sagte sagen VVFIN
 267034 Mittelfrankens Mittelfranken NE
 267035 Polizeipr\"asident Polizeipr\"asident NN
 267036 Peter Peter NE
 267037 von von NE
 267038 der der NE
 267039 Grn Grn NE

vs. Tagger

sagte VVFIN sagen
 Mittelfrankens NE Mittelfranken
 Polizeiprsident NN Polizeiprsident
 Peter NE Peter
 von APPR von
 der ART d
 Gr\"un NN Gr\"un

Here the whole multiword expression is tagged as NE in the gold standard corpus which is surely unfeasible for automatic taggers. In order to avoid such errors, it is necessary to tokenize such cases appropriately as multiword units and not as single independent words before doing automatic POS-tagging. On the other side, the question is whether it is more appropriate to perform this step before tagging, or whether it would make sense to tag such cases word-by-word and to find en-

tity chunks afterwards. This problem is similar to the situation with proper and common nouns: the question is always, whether it makes sense to give over this task to the tagger or to catch up the differentiation in the further processing steps. The latter seems to be more reasonable.

This type of tagger deficiencies can be supported with better *rule-based* after processing, as rules are good at disambiguating such trivial and often naive errors that pose a problem for a HMM tagger because of its locally non-distinctive behaviour.

2. errors and inconsistencies in the reference corpus, that make the evaluation of the tagger's true accuracy almost impossible as well as introduce noise into the trained models;

e.g.

```
\$ grep -w "Vatikan" TigerGold
```

```
Vatikan NE Vatikan
```

```
Vatikan NN Vatikan
```

```
Vatikan NE Vatikan
```

```
Vatikan NE Vatikan
```

Actually, 98% accuracy would not be that bad and it is probably unfeasable to come over this limit, as it correlates more or less with the inter-annotator agreement of 98-99%. However, it has been shown in this thesis that even accuracies of 96%-98% are still unreachable for web as corpus.

The main contribution of this work was the evaluation of part-of-speech tagging for web as corpus, which is the first evaluation of this kind, as to our knowledge. For this, a section of DEWAC corpus was semi-automatically annotated, i.e. it was first preannotated by the TREE TAGGER and then manually corrected.

Three state-of-the-art taggers have been evaluated on this subcorpus and compared to the analogous evaluation under the exactly same conditions for a more traditional newspaper corpus.

The accuracy of the `TREETAGGER`, which is the most widely used tagger for German, on the web corpus, provided that the text was correctly tokenized, was about 93.7%. This is ca. 4% less than the reported one for this tagger. However, the text level evaluation of `DEWAC` proved, that most traditional texts on the web can be tagged rather well (at about 96%-97% accuracy), while some other cases, namely the representative genres of web 2.0 or collaborative web, are disastrous. One could remove such texts from the corpus if they are irrelevant for particular research or application purposes, or focus on recognising them and activating special heuristics or different tagging models for these genres. Also, a better treatment of unknowns, as well as symbols prevalent on web (i.e. specific punctuation, foreign material, encodings) is crucial.

References

- Apache, Software Foundation. UIMA Sandbox Tagger. [41](#)
- Baroni, M., and A. Kilgarriff. 2006. Large linguistically-processed web corpora for multiple languages. In *Proceedings of EACL*, 87–90. [44](#)
- Biemann, C. 2006a. Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of the HLT-NAACL-06 Workshop on Textgraphs-06*. New York, USA. [33](#)
- . 2006b. Unsupervised Part-of-Speech Tagging Employing Efficient Graph Clustering. In *Proceedings of the COLING/ACL-06 Student Research Workshop 2006*. Sydney, Australia. [33](#)
- Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*. Sozopol. [2](#), [45](#)
- Brants, Sabine, and Silvia Hansen. 2002. Developments in the Tiger Annotation Scheme and their Realization in the Corpus. In *Proceedings of the Third Conference on Language Resources and Evaluation (LREC 2002)*, 1643–1649. [45](#)
- Brants, Thorsten. 2000. TnT – a statistical part-of-speech tagger. In *Anlp*, 224–231. [i](#), [1](#), [9](#), [13](#), [14](#), [18](#), [19](#), [34](#), [37](#), [40](#), [41](#), [46](#)
- Brill, Eric. 1994. Some advances in transformation-based part of speech tagging. In *National Conference on Artificial Intelligence*, 722–727. [iv](#), [15](#), [16](#)

REFERENCES

- . 1995. Unsupervised learning of disambiguation rules for part of speech tagging. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, 1–13. [iv](#), [4](#), [12](#), [15](#), [16](#), [17](#), [18](#), [37](#)
- Brill, Erik, and Jun Wu. 1998. Classifier combination for improved lexical disambiguation. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association of Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, ed. Christian Boitet and Pete Whitelock, 191–195. San Francisco, California: Morgan Kaufmann Publishers. [iv](#), [22](#), [26](#), [27](#), [28](#), [30](#)
- Charniak, Eugene, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz. 1993. Equations for part-of-speech tagging. In *National Conference on Artificial Intelligence*, 784–789. [4](#)
- Chowdhury, Abdur, and M. Catherine McCabe. 1993. Improving information retrieval systems using parts of speech tagging. Tech. Rep. [1](#)
- Christianini, N., and J. Shawe-Taylor, eds. 2000. *An Introduction to Support Vector Machines*. Cambridge University Press. [33](#)
- Church, Kenneth. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference of Applied Natural Language Processing*, vol. 136. [i](#), [8](#), [18](#)
- . 1992. Current practice in part of speech tagging and suggestions for the future. In *Sbornik Praci: In Honour of Henry Ku'cera*, ed. Simmons, 13–48. University of Michigan: Michigan Slavic Studies. [1](#), [35](#)
- Cohen, Jacob. 1960. A coefficient of agreement for nominal scales. In *Educational and psychological measurement*, vol. 20, 37–46. [58](#)
- Cutting, Doug, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*. [22](#)

REFERENCES

- Dang, Hoa Trang, and Martha Palmer. 2002. Combining contextual features for word sense disambiguation. In *Proceedings of the Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, 88–94. 1
- EAGLES. Version of May, 1996. Study of the relation between tagsets and taggers. EAGLES Validation Document (EAG-CLWG-TAGS/V). 39, 42, 44
- Francis, W. Nelson, and Henry Kucera. 1982. *Frequency analysis of English usage. Lexicon and grammar*. Houghton Mifflin, Boston. 4, 8, 17, 23
- Garside, Geoffrey Leech, Roger, and Geoffrey Sampson, eds. 1987. *The computational analysis of English: A corpus-based approach*. Harlow: Longman. 42
- Gimenez, D., and L. Marquez. 2004. Svmtool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of the Fourth Conference on Language Resources and Evaluation*. Lisbon, Portugal. iv, 33, 37
- Glickman, Oren, and Rosie Jones. 1999. Examining machine learning for adaptable end-to-end information extraction systems. In *AAAI Workshop on Machine Learning for Information Extraction*. 1
- Greene, B.B., and G.M. Rubin. 1971. Automatic grammatical tagging of English. technical report, Department of Linguistics, Brown University, Providence, Rhode Island. 4
- Halteren, Hans van, Walter Daelemans, and Jakub Zavrel. 2001. Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics* 27(27(2)):199–230. iv, 2, 29, 30, 35
- Harris, Z. 1968. *Mathematical Structures of Language*. New York: Wiley & Sons. 31
- Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A German computational morphology covering derivation, composition and inflection. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, vol. 4, 1263–1266. Lisbon, Portugal. 40

REFERENCES

- Järvinen, Timo. 1994. Annotating 200 million words: The Bank of English project. In *Proceedings of COLING-94*, vol. I, 565–568. Kyoto, Japan. [23](#)
- Jiménez, D., E. Ferretti, V. Vidal, P. Rosso, C. F. Enguix, and C. F. Enguix. 2003. The influence of semantics in IR using LSI and K-means clustering techniques. In *ISICT '03: Proceedings of the 1st international symposium on information and communication technologies*, 279–284. Trinity College Dublin. [1](#)
- Johansson, S. 1986. *The tagged LOB Corpus: User's Manual*. Norwegian Computing Center for the Humanities, Bergen, Norway. [30](#)
- Karlsson, Voutilainen A. Heikkilä J. & Antilla A., F., ed. 1995. *Constraint grammar - a language-independent system for parsing unrestricted text*. Mouton de Gruyter, Berlin. [22](#)
- MacKinlay, Andrew, and Timothy Baldwin. 2005. POS tagging with a more informative tagset. In *Proceedings of the Australian Language Technology Workshop*, 40–48. Sydney, Australia. [22](#), [35](#)
- Manning, Christopher D., and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press. [5](#), [6](#), [7](#), [42](#)
- Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330. [12](#), [14](#), [17](#), [34](#), [45](#)
- Oy, Lingsoft. 1994. Gertwol, questionnaire for morpholympics. *LDV-Forum* 11(1): 17–29. [28](#)
- Padro, Lluís, and Lluís Marquez. 1998. On the evaluation and comparison of taggers: the effect of noise in testing corpora. In *COLING-ACL*, 997–1002. [35](#)
- Ramshaw, Lance, and Mitchell Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the ACL Third Workshop on Very Large Corpora*, 82–94. [15](#)

REFERENCES

- Ratnaparkhi, Adwait. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing*, 133–142. University of Pennsylvania. [iv](#), [18](#), [19](#), [20](#), [21](#), [22](#), [35](#), [37](#)
- Samuel, Sandra Carberry, Ken, and K. Vijay-Shanker. 1998. Dialogue act tagging with transformation-based learning. In *Proceedings of COLING/ACL'98*, 1150–1156. [15](#)
- Schiller, A. 1995. DMOR:Benutzeranleitung. Tech. Rep. [12](#)
- Schmid, Helmut. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*. [i](#), [9](#), [29](#)
- . 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*. [iii](#), [iv](#), [1](#), [9](#), [10](#), [12](#), [13](#), [32](#), [37](#), [45](#), [48](#), [49](#), [52](#), [62](#), [65](#)
- Schütze, Hinrich. 1995. Distributional part of speech tagging. *EACL* . [31](#), [32](#), [33](#)
- Tapanainen, Pasi, and Atro Voutilainen. 1994. Tagging accurately – don't guess if you know. In *Proceedings of the Fourth ACL Conference on Applied Natural Language*. Stuttgart. [i](#), [22](#), [23](#), [25](#), [26](#), [34](#), [35](#), [73](#)
- Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*. [1](#), [22](#), [33](#)
- Volk, Martin, and Gerold Schneider. 1998. Comparing a statistical and a rule-based tagger for German. In *Computers, Linguistics, and Phonetics between Language and Speech. Proceedings of the 4th Conference on Natural Language Processing*, 125–137. KONVENS-98, Bonn. [iv](#), [28](#), [29](#), [30](#), [37](#), [52](#), [57](#), [65](#)
- Zipf, G.K. 1935. *The Psycho-Biology of Language*. Houghton Mifflin Co. [8](#)

List of Abbreviations

Roman Symbols

CV	Cross Validation
DE	Disambiguation Error
EAGLES	Expert Advisory Group on Language Engineering Standards
HMM	Hidden Markov Model
LE	Lexical Error
LOB	Lancaster-Oslo/Bergen corpus
MaxEnt	Maximum Entropy
SPF	Standard Parameter File
WSJ	Wall Street Journal