TnT — A Statistical Part-of-Speech Tagger

Thorsten Brants
Saarland University
Computational Linguistics
D-66041 Saarbrücken, Germany
thorsten@coli.uni-sb.de

Abstract

Trigrams'n'Tags (TnT) is an efficient statistical part-of-speech tagger. Contrary to claims found elsewhere in the literature, we argue that a tagger based on Markov models performs at least as well as other current approaches, including the Maximum Entropy framework. A recent comparison has even shown that TnT performs significantly better for the tested corpora. We describe the basic model of TnT, the techniques used for smoothing and for handling unknown words. Furthermore, we present evaluations on two corpora.

1 Introduction

A large number of current language processing systems use a part-of-speech tagger for pre-processing. The tagger assigns a (unique or ambiguous) part-of-speech tag to each token in the input and passes its output to the next processing level, usually a parser. Furthermore, there is a large interest in part-of-speech tagging for corpus annotation projects, who create valuable linguistic resources by a combination of automatic processing and human correction.

For both applications, a tagger with the highest possible accuracy is required. The debate about which paradigm solves the part-of-speech tagging problem best is not finished. Recent comparisons of approaches that can be trained on corpora (van Halteren et al., 1998; Volk and Schneider, 1998) have shown that in most cases statistical aproaches (Cutting et al., 1992; Schmid, 1995; Ratnaparkhi, 1996) yield better results than finite-state, rule-based, or memory-based taggers (Brill, 1993; Daelemans et al., 1996). They are only surpassed by combinations of different systems, forming a "voting tagger".

Among the statistical approaches, the Maximum Entropy framework has a very strong position. Nevertheless, a recent independent comparison of 7 taggers (Zavrel and Daelemans, 1999) has shown that another approach even works better: Markov models combined with a good smoothing technique and with handling of unknown words. This tagger, TnT, not only yielded the highest accuracy, it also was the fastest both in training and tagging.

The tagger comparison was organized as a "black-box test": set the same task to every tagger and compare the outcomes. This paper describes the models and techniques used by TnT together with the implementation.

The reader will be surprised how simple the underlying model is. The result of the tagger comparison seems to support the maxime "the simplest is the best". However, in this paper we clarify a number of details that are omitted in major previous publications concerning tagging with Markov models. As two examples, (Rabiner, 1989) and (Charniak et al., 1993) give good overviews of the techniques and equations used for Markov models and part-of-speech tagging, but they are not very explicit in the details that are needed for their application. We argue that it is not only the choice of the general model that determines the result of the tagger but also the various "small" decisions on alternatives.

The aim of this paper is to give a detailed account of the techniques used in TnT. Additionally, we present results of the tagger on the NEGRA corpus (Brants et al., 1999) and the Penn Treebank (Marcus et al., 1993). The Penn Treebank results reported here for the Markov model approach are at least equivalent to those reported for the Maximum Entropy approach in (Ratnaparkhi, 1996). For a comparison to other taggers, the reader is referred to (Zavrel and Daelemans, 1999).

2 Architecture

2.1 The Underlying Model

TnT uses second order Markov models for part-of-speech tagging. The states of the model represent tags, outputs represent the words. Transition probabilities depend on the states, thus pairs of tags. Output probabilities only depend on the most recent category. To be explicit, we calculate

$$\underset{t_1 \ldots t_T}{\operatorname{argmax}} \left[\prod_{i=1}^{T} P(t_i|t_{i-1}, t_{i-2}) P(w_i|t_i) \right] P(t_{T+1}|t_T) \tag{1}$$

for a given sequence of words $w_1 \ldots w_T$ of length T. $t_1 \ldots t_T$ are elements of the tagset, the additional

tags t_{-1}, t_0, and t_{T+1} are beginning-of-sequence and end-of-sequence markers. Using these additional tags, even if they stem from rudimentary processing of punctuation marks, slightly improves tagging results. This is different from formulas presented in other publications, which just stop with a "loose end" at the last word. If sentence boundaries are not marked in the input, TnT adds these tags if it encounters one of [.!?;] as a token.

Transition and output probabilities are estimated from a tagged corpus. As a first step, we use the maximum likelihood probabilities $\hat{P}$ which are derived from the relative frequencies:

$$\text{Unigrams:} \quad \hat{P}(t_3) = \frac{f(t_3)}{N} \tag{2}$$

$$\text{Bigrams:} \quad \hat{P}(t_3|t_2) = \frac{f(t_2,t_3)}{f(t_2)} \tag{3}$$

$$\text{Trigrams:} \quad \hat{P}(t_3|t_1,t_2) = \frac{f(t_1,t_2,t_3)}{f(t_1,t_2)} \tag{4}$$

$$\text{Lexical:} \quad \hat{P}(w_3|t_3) = \frac{f(w_3,t_3)}{f(t_3)} \tag{5}$$

for all t_1, t_2, t_3 in the tagset and w_3 in the lexicon. N is the total number of tokens in the training corpus. We define a maximum likelihood probability to be zero if the corresponding nominators and denominators are zero. As a second step, contextual frequencies are smoothed and lexical frequences are completed by handling words that are not in the lexicon (see below).

2.2 Smoothing

Trigram probabilities generated from a corpus usually cannot directly be used because of the sparse-data problem. This means that there are not enough instances for each trigram to reliably estimate the probability. Furthermore, setting a probability to zero because the corresponding trigram never occured in the corpus has an undesired effect. It causes the probability of a complete sequence to be set to zero if its use is necessary for a new text sequence, thus makes it impossible to rank different sequences containing a zero probability.

The smoothing paradigm that delivers the best results in TnT is linear interpolation of unigrams, bigrams, and trigrams. Therefore, we estimate a trigram probability as follows:

$$P(t_3|t_1,t_2) = \lambda_1\hat{P}(t_3) + \lambda_2\hat{P}(t_3|t_2) + \lambda_3\hat{P}(t_3|t_1,t_2) \tag{6}$$

$\hat{P}$ are maximum likelihood estimates of the probabilities, and $\lambda_1 + \lambda_2 + \lambda_3 = 1$, so P again represent probability distributions.

We use the context-independent variant of linear interpolation, i.e., the values of the λs do not depend on the particular trigram. Contrary to intuition, this yields better results than the context-dependent variant. Due to sparse-data problems, one cannot estimate a different set of λs for each trigram. Therefore, it is common practice to group trigrams by frequency and estimate tied sets of λs. However, we are not aware of any publication that has investigated frequency groupings for linear interpolation in part-of-speech tagging. All groupings that we have tested yielded at most equivalent results to context-independent linear interpolation. Some groupings even yielded worse results. The tested groupings included a) one set of λs for each frequency value and b) two classes (low and high frequency) on the two ends of the scale, as well as several groupings in between and several settings for partitioning the classes.

The values of λ_1, λ_2, and λ_3 are estimated by deleted interpolation. This technique successively removes each trigram from the training corpus and estimates best values for the λs from all other n-grams in the corpus. Given the frequency counts for uni-, bi-, and trigrams, the weights can be very efficiently determined with a processing time linear in the number of different trigrams. The algorithm is given in figure 1. Note that subtracting 1 means taking unseen data into account. Without this subtraction the model would overfit the training data and would generally yield worse results.

2.3 Handling of Unknown Words

Currently, the method of handling unknown words that seems to work best for inflected languages is a suffix analysis as proposed in (Samuelsson, 1993). Tag probabilities are set according to the word's ending. The suffix is a strong predictor for word classes, e.g., words in the Wall Street Journal part of the Penn Treebank ending in *able* are adjectives (JJ) in 98% of the cases (e.g. fashionable, variable) , the rest of 2% are nouns (e.g. cable, variable).

The probability distribution for a particular suffix is generated from all words in the training set that share the same suffix of some predefined maximum length. The term suffix as used here means "final sequence of characters of a word" which is not necessarily a linguistically meaningful suffix.

Probabilities are smoothed by successive abstraction. This calculates the probability of a tag t given the last m letters l_i of an n letter word: $P(t|l_{n-m+1},\ldots l_n)$. The sequence of increasingly more general contexts omits more and more characters of the suffix, such that $P(t|l_{n-m+2},\ldots,l_n)$, $P(t|l_{n-m+3},\ldots,l_n)$, $\ldots$, $P(t)$ are used for smoothing. The recursion formula is

$$P(t|l_{n-i+1},\ldots l_n) = \frac{\hat{P}(t|l_{n-i+1},\ldots l_n) + \theta_i P(t|l_{n-i},\ldots,l_n)}{1+\theta_i} \tag{7}$$

```
set λ1 = λ2 = λ3 = 0
foreach trigram t1,t2,t3 with f(t1,t2,t3) > 0
    depending on the maximum of the following three values:
        case (f(t1,t2,t3)-1)/(f(t1,t2)-1): increment λ3 by f(t1,t2,t3)
        case (f(t2,t3)-1)/(f(t2)-1): increment λ2 by f(t1,t2,t3)
        case (f(t3)-1)/(N-1): increment λ1 by f(t1,t2,t3)
    end
end
normalize λ1,λ2,λ3
```

Figure 1: Algorithm for calculting the weights for context-independent linear interpolation $\lambda_1, \lambda_2, \lambda_3$ when the n-gram frequencies are known. N is the size of the corpus. If the denominator in one of the expressions is 0, we define the result of that expression to be 0.

for $i = m \ldots 0$, using the maximum likelihood estimates $\hat{P}$ from frequencies in the lexicon, weights θ_i and the initialization

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

The maximum likelihood estimate for a suffix of length i is derived from corpus frequencies by

$$\hat{P}(t|l_{n-i+1}, \ldots l_n) = \frac{f(t, l_{n-i+1}, \ldots l_n)}{f(l_{n-i+1}, \ldots l_n)} \tag{9}$$

For the Markov model, we need the inverse conditional probabilities $P(l_{n-i+1}, \ldots l_n|t)$ which are obtained by Bayesian inversion.

A theoretical motivated argumentation uses the standard deviation of the maximum likelihood probabilities for the weights θ_i (Samuelsson, 1993).

This leaves room for interpretation.

1) One has to identify a good value for m, the longest suffix used. The approach taken for TnT is the following: m depends on the word in question. We use the longest suffix that we can find in the training set (i.e., for which the frequency is greater than or equal to 1), but at most 10 characters. This is an empirically determined choice.

2) We use a context-independent approach for θ_i, as we did for the contextual weights λ_i. It turned out to be a good choice to set all θ_i to the standard deviation of the unconditioned maximum likelihood probabilities of the tags in the training corpus, i.e., we set

$$\theta_i = \frac{1}{s-1} \sum_{j=1}^{s} (\hat{P}(t_j) - \bar{P})^2 \tag{10}$$

for all $i = 0 \ldots m-1$, using a tagset of s tags and the average

$$\bar{P} = \frac{1}{s} \sum_{j=1}^{s} \hat{P}(t_j) \tag{11}$$

This usually yields values in the range $0.03 \ldots 0.10$.

3) We use different estimates for uppercase and lowercase words, i.e., we maintain two different suffix tries depending on the capitalization of the word. This information improves the tagging results.

4) Another freedom concerns the choice of the words in the lexicon that should be used for suffix handling. Should we use all words, or are some of them better suited than others? Accepting that unknown words are most probably infrequent, one can argue that using suffixes of infrequent words in the lexicon is a better approximation for unknown words than using suffixes of frequent words. Therefore, we restrict the procedure of suffix handling to words with a frequency smaller than or equal to some threshold value. Empirically, 10 turned out to be a good choice for this threshold.

2.4 Capitalization

Additional information that turned out to be useful for the disambiguation process for several corpora and tagsets is capitalization information. Tags are usually not informative about capitalization, but probability distributions of tags around capitalized words are different from those not capitalized. The effect is larger for English, which only capitalizes proper names, and smaller for German, which capitalizes all nouns.

We use flags c_i that are true if w_i is a capitalized word and false otherwise. These flags are added to the contextual probability distributions. Instead of

$$P(t_3|t_1, t_2) \tag{12}$$

we use

$$P(t_3, c_3|t_1, c_1, t_2, c_2) \tag{13}$$

and equations (3) to (5) are updated accordingly. This is equivalent to doubling the size of the tagset and using different tags depending on capitalization.

2.5 Beam Search

The processing time of the Viterbi algorithm (Rabiner, 1989) can be reduced by introducing a beam search. Each state that receives a δ value smaller than the largest δ divided by some threshold value θ is excluded from further processing. While the Viterbi algorithm is guaranteed to find the sequence of states with the highest probability, this is no longer true when beam search is added. Nevertheless, for practical purposes and the right choice of θ, there is virtually no difference between the algorithm with and without a beam. Empirically, a value of $\theta = 1000$ turned out to approximately double the speed of the tagger without affecting the accuracy.

The tagger currently tags between 30,000 and 60,000 tokens per second (including file I/O) on a Pentium 500 running Linux. The speed mainly depends on the percentage of unknown words and on the average ambiguity rate.

3 Evaluation

We evaluate the tagger's performance under several aspects. First of all, we determine the tagging accuracy averaged over ten iterations. The overall accuracy, as well as separate accuracies for known and unknown words are measured.

Second, learning curves are presented, that indicate the performance when using training corpora of different sizes, starting with as few as 1,000 tokens and ranging to the size of the entire corpus (minus the test set).

An important characteristic of statistical taggers is that they not only assign tags to words but also probabilities in order to rank different assignments. We distinguish reliable from unreliable assignments by the quotient of the best and second best assignments[1]. All assignments for which this quotient is larger than some threshold are regarded as reliable, the others as unreliable. As we will see below, accuracies for reliable assignments are much higher.

The tests are performed on partitions of the corpora that use 90% as training set and 10% as test set, so that the test data is guaranteed to be unseen during training. Each result is obtained by repeating the experiment 10 times with different partitions and averaging the single outcomes.

In all experiments, contiguous test sets are used. The alternative is a round-robin procedure that puts every 10th sentence into the test set. We argue that contiguous test sets yield more realistic results because completely unseen articles are tagged. Using the round-robin procedure, parts of an article are already seen, which significantly reduces the percentage of unknown words. Therefore, we expect even higher results when testing on every 10th sentence instead of a contiguous set of 10%.

In the following, accuracy denotes the number of correctly assigned tags divided by the number of tokens in the corpus processed. The tagger is allowed to assign exactly one tag to each token.

We distinguish the overall accuracy, taking into account all tokens in the test corpus, and separate accuracies for known and unknown tokens. The latter are interesting, since usually unknown tokens are much more difficult to process than known tokens, for which a list of valid tags can be found in the lexicon.

3.1 Tagging the NEGRA corpus

The German NEGRA corpus consists of 20,000 sentences (355,000 tokens) of newspaper texts (Frankfurter Rundschau) that are annotated with parts-of-speech and predicate-argument structures (Skut et al., 1997). It was developed at the Saarland University in Saarbrücken[2]. Part of it was tagged at the IMS Stuttgart. This evaluation only uses the part-of-speech annotation and ignores structural annotations.

Tagging accuracies for the NEGRA corpus are shown in table 2.

Figure 3 shows the learning curve of the tagger, i.e., the accuracy depending on the amount of training data. Training length is the number of tokens used for training. Each training length was tested ten times, training and test sets were randomly chosen and disjoint, results were averaged. The training length is given on a logarithmic scale.

It is remarkable that tagging accuracy for known words is very high even for very small training corpora. This means that we have a good chance of getting the right tag if a word is seen at least once during training. Average percentages of unknown tokens are shown in the bottom line of each diagram.

We exploit the fact that the tagger not only determines tags, but also assigns probabilities. If there is an alternative that has a probability "close to" that of the best assignment, this alternative can be viewed as almost equally well suited. The notion of "close to" is expressed by the distance of probabilities, and this in turn is expressed by the quotient of probabilities. So, the distance of the probabilities of a best tag t_{best} and an alternative tag t_{alt} is expressed by $p(t_{best})/p(t_{alt})$, which is some value greater or equal to 1 since the best tag assignment has the highest probability.

Figure 4 shows the accuracy when separating assignments with quotients larger and smaller than the threshold (hence reliable and unreliable assignments). As expected, we find that accuracies for

[1] By definition, this quotient is ∞ if there is only one possible tag for a given word.

[2] For availability, please check `http://www.coli.uni-sb.de/sfb378/negra-corpus`

Table 2: Part-of-speech tagging accuracy for the NEGRA corpus, averaged over 10 test runs, training and test set are disjoint. The table shows the percentage of unknown tokens, separate accuracies and standard deviations for known and unknown tokens, as well as the overall accuracy.

	percentage unknowns	known acc.	σ	unknown acc.	σ	overall acc.	σ
NEGRA corpus	11.9%	97.7%	0.23	89.0%	0.72	96.7%	0.29

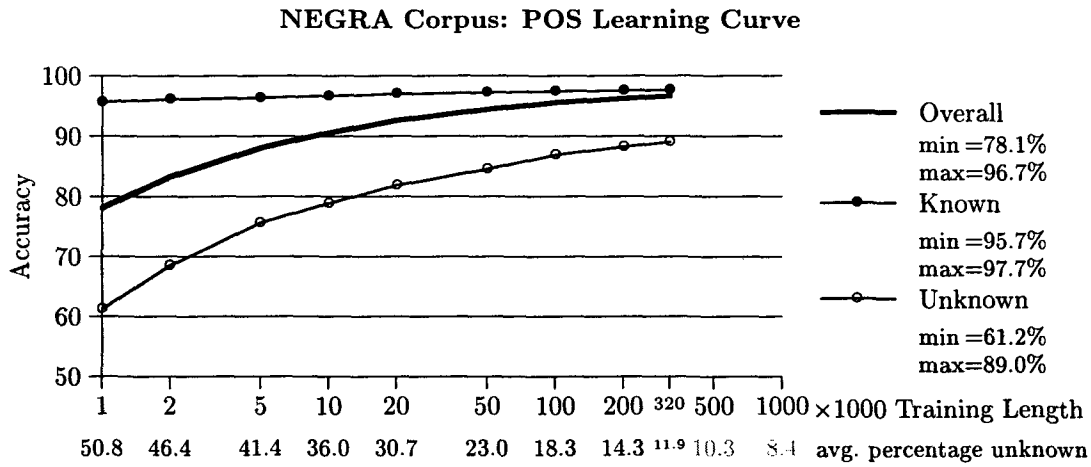

Figure 3: Learning curve for tagging the NEGRA corpus. The training sets of variable sizes as well as test sets of 30,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged. Percentages of unknowns for 500k and 1000k training are determined from an untagged extension.

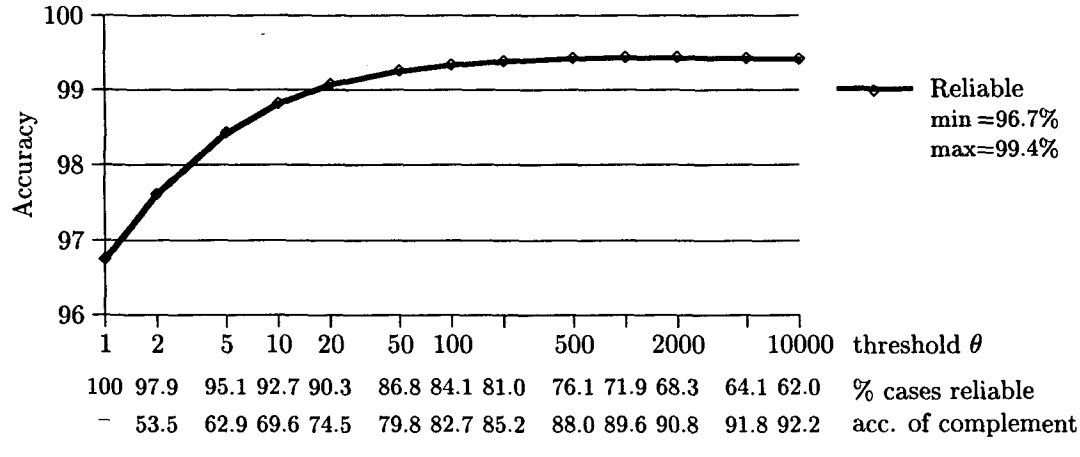

Figure 4: Tagging accuracy for the NEGRA corpus when separating reliable and unreliable assignments. The curve shows accuracies for reliable assignments. The numbers at the bottom line indicate the percentage of reliable assignments and the accuracy of the complement set (i.e., unreliable assignments).

Table 5: Part-of-speech tagging accuracy for the Penn Treebank. The table shows the percentage of unknown tokens, separate accuracies and standard deviations for known and unknown tokens, as well as the overall accuracy.

	percentage unknowns	known acc.	known σ	unknown acc.	unknown σ	overall acc.	overall σ
Penn Treebank	2.9%	97.0%	0.15	85.5%	0.69	96.7%	0.15

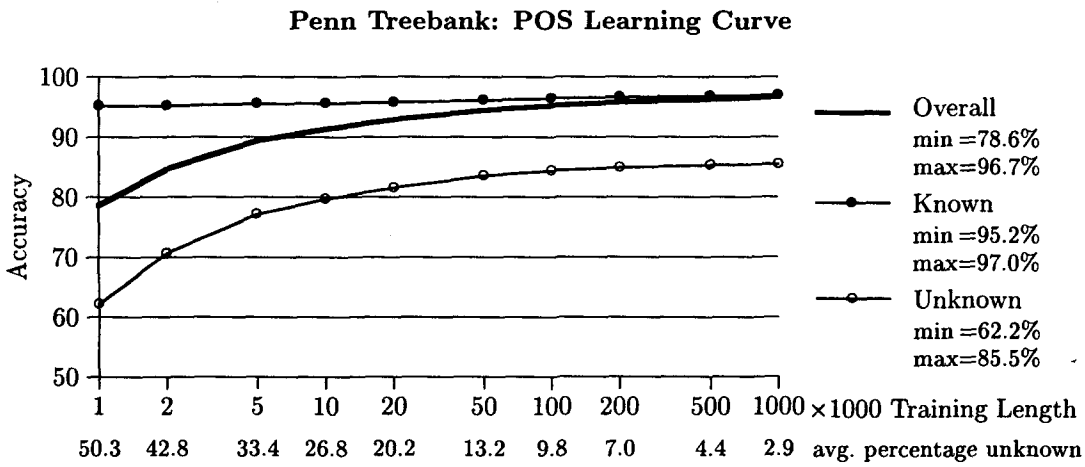

Figure 6: Learning curve for tagging the Penn Treebank. The training sets of variable sizes as well as test sets of 100,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

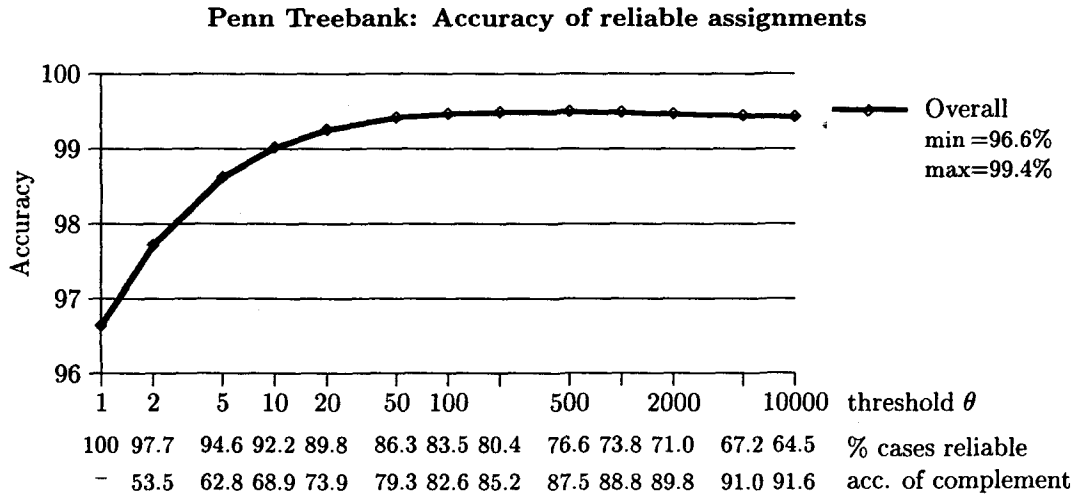

Figure 7: Tagging accuracy for the Penn Treebank when separating reliable and unreliable assignments. The curve shows accuracies for reliable assignments. The numbers at the bottom line indicate the percentage of reliable assignments and the accuracy of the complement set.

reliable assignments are much higher than for unreliable assignments. This distinction is, e.g., useful for annotation projects during the cleaning process, or during pre-processing, so the tagger can emit multiple tags if the best tag is classified as unreliable.

3.2 Tagging the Penn Treebank

We use the Wall Street Journal as contained in the Penn Treebank for our experiments. The annotation consists of four parts: 1) a context-free structure augmented with traces to mark movement and discontinuous constituents, 2) phrasal categories that are annotated as node labels, 3) a small set of grammatical functions that are annotated as extensions to the node labels, and 4) part-of-speech tags (Marcus et al., 1993). This evaluation only uses the part-of-speech annotation.

The Wall Street Journal part of the Penn Treebank consists of approx. 50,000 sentences (1.2 million tokens).

Tagging accuracies for the Penn Treebank are shown in table 5. Figure 6 shows the learning curve of the tagger, i.e., the accuracy depending on the amount of training data. Training length is the number of tokens used for training. Each training length was tested ten times. Training and test sets were disjoint, results are averaged. The training length is given on a logarithmic scale. As for the NEGRA corpus, tagging accuracy is very high for known tokens even with small amounts of training data.

We exploit the fact that the tagger not only determines tags, but also assigns probabilities. Figure 7 shows the accuracy when separating assignments with quotients larger and smaller than the threshold (hence reliable and unreliable assignments). Again, we find that accuracies for reliable assignments are much higher than for unreliable assignments.

3.3 Summary of Part-of-Speech Tagging Results

Average part-of-speech tagging accuracy is between 96% and 97%, depending on language and tagset, which is at least on a par with state-of-the-art results found in the literature, possibly better. For the Penn Treebank, (Ratnaparkhi, 1996) reports an accuracy of 96.6% using the Maximum Entropy approach, our much simpler and therefore faster HMM approach delivers 96.7%. This comparison needs to be re-examined, since we use a ten-fold crossvalidation and averaging of results while Ratnaparkhi only makes one test run.

The accuracy for known tokens is significantly higher than for unknown tokens. For the German newspaper data, results are 8.7% better when the word was seen before and therefore is in the lexicon, than when it was not seen before (97.7% vs. 89.0%). Accuracy for known tokens is high even with very small amounts of training data. As few as 1000 tokens are sufficient to achieve 95%–96% accuracy for them. It is important for the tagger to have seen a word at least once during training.

Stochastic taggers assign probabilities to tags. We exploit the probabilities to determine reliability of assignments. For a subset that is determined during processing by the tagger we achieve accuracy rates of over 99%. The accuracy of the complement set is much lower. This information can, e.g., be exploited in an annotation project to give an additional treatment to the unreliable assignments, or to pass selected ambiguities to a subsequent processing step.

4 Conclusion

We have shown that a tagger based on Markov models yields state-of-the-art results, despite contrary claims found in the literature. For example, the Markov model tagger used in the comparison of (van Halteren et al., 1998) yielded worse results than all other taggers. In our opinion, a reason for the wrong claim is that the basic algorithms leave several decisions to the implementor. The rather large amount of freedom was not handled in detail in previous publications: handling of start- and end-of-sequence, the exact smoothing technique, how to determine the weights for context probabilities, details on handling unknown words, and how to determine the weights for unknown words. Note that the decisions we made yield good results for both the German and the English Corpus. They do so for several other corpora as well. The architecture remains applicable to a large variety of languages.

According to current tagger comparisons (van Halteren et al., 1998; Zavrel and Daelemans, 1999), and according to a comparsion of the results presented here with those in (Ratnaparkhi, 1996), the Maximum Entropy framework seems to be the only other approach yielding comparable results to the one presented here. It is a very interesting future research topic to determine the advantages of either of these approaches, to find the reason for their high accuracies, and to find a good combination of both.

TnT is freely available to universities and related organizations for research purposes (see `http://www.coli.uni-sb.de/~thorsten/tnt`).

Acknowledgements

Many thanks go to Hans Uszkoreit for his support during the development of TnT. Most of the work on TnT was carried out while the author received a grant of the Deutsche Forschungsgemeinschaft in the Graduiertenkolleg Kognitionswissenschaft Saarbrücken. Large annotated corpora are the pre-requisite for developing and testing part-of-speech taggers, and they enable the generation of high-quality language models. Therefore, I would

like to thank all the people who took the effort to annotate the Penn Treebank, the Susanne Corpus, the Stuttgarter Referenzkorpus, the NEGRA Corpus, the Verbmobil Corpora, and several others. And, last but not least, I would like to thank the users of TnT who provided me with bug reports and valuable suggestions for improvements.

References

Thorsten Brants, Wojciech Skut, and Hans Uszkoreit. 1999. Syntactic annotation of a German newspaper corpus. In *Proceedings of the ATALA Treebank Workshop*, pages 69–76, Paris, France.

Eric Brill. 1993. *A Corpus-Based Approach to Language Learning.* Ph.D. Dissertation, Department of Computer and Information Science, University of Pennsylvania.

Eugene Charniak, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz. 1993. Equations for part-of-speech tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789, Menlo Park: AAAI Press/MIT Press.

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the 3rd Conference on Applied Natural Language Processing (ACL)*, pages 133–140.

Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven Gillis. 1996. Mbt: A memory-based part of speech tagger-generator. In *Proceedings of the Workshop on Very Large Corpora*, Copenhagen, Denmark.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Lawrence R. Rabiner. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77(2), pages 257–285.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing EMNLP-96*, Philadelphia, PA.

Christer Samuelsson. 1993. Morphological tagging based entirely on Bayesian inference. In *9th Nordic Conference on Computational Linguistics NODALIDA-93*, Stockholm University, Stockholm, Sweden.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In Helmut Feldweg and Erhard Hinrichts, editors, *Lexikon und Text.* Niemeyer, Tübingen.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing ANLP-97*, Washington, DC.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 1998. Improving data driven wordclass tagging by system combination. In *Proceedings of the International Conference on Computational Linguistics COLING-98*, pages 491–497, Montreal, Canada.

Martin Volk and Gerold Schneider. 1998. Comparing a statistical and a rule-based tagger for german. In *Proceedings of KONVENS-98*, pages 125–137, Bonn.

Jakub Zavrel and Walter Daelemans. 1999. Evaluatie van part-of-speech taggers voor het corpus gesproken nederlands. CGN technical report, Katholieke Universiteit Brabant, Tilburg.