Estimation of Conditional Probabilities With Decision Trees and an Application to Fine-Grained POS Tagging

Helmut Schmid and Florian Laws
IMS, University of Stuttgart
{schmid,lawsfn}@ims.uni-stuttgart.de

Abstract

We present a HMM part-of-speech tagging method which is particularly suited for POS tagsets with a large number of fine-grained tags. It is based on three ideas: (1) splitting of the POS tags into attribute vectors and decomposition of the contextual POS probabilities of the HMM into a product of attribute probabilities, (2) estimation of the contextual probabilities with decision trees, and (3) use of high-order HMMs. In experiments on German and Czech data, our tagger outperformed state-of-the-art POS taggers.

1 Introduction

A Hidden-Markov-Model part-of-speech tagger (Brants, 2000, e.g.) computes the most probable POS tag sequence $\hat{t}_1^N = \hat{t}_1, ..., \hat{t}_N$ for a given word sequence w_1^N.

$$\hat{t}_1^N = \arg\max_{t_1^N} p(t_1^N, w_1^N)$$

The joint probability of the two sequences is defined as the product of context probabilities and lexical probabilities over all POS tags:

$$p(t_1^N, w_1^N) = \prod_{i=1}^{N} \underbrace{p(t_i|t_{i-k}^{i-1})}_{\text{context prob.}} \underbrace{p(w_i|t_i)}_{\text{lexical prob.}} \quad (1)$$

HMM taggers are fast and were successfully applied to a wide range of languages and training corpora.

POS taggers are usually trained on corpora with between 50 and 150 different POS tags. Tagsets of this size contain little or no information about number, gender, case and similar morphosyntactic features. For languages with a rich morphology such as German or Czech, more fine-grained tagsets are often considered more appropriate. The additional information may also help to disambiguate the (base) part of speech. Without gender information, for instance, it is difficult for a tagger to correctly disambiguate the German sentence *Ist das Realität?* (Is that reality?). The word *das* is ambiguous between an article and a demonstrative. Because of the lack of gender agreement between *das* (neuter) and the noun *Realität* (feminine), the article reading must be wrong.

The German Tiger treebank (Brants et al., 2002) is an example of a corpus with a more fine-grained tagset (over 700 tags overall). Large tagsets aggravate sparse data problems. As an example, take the German sentence *Das zu versteuernde Einkommen sinkt* ("The to be taxed income decreases"; The taxable income decreases). This sentence should be tagged as shown in table 1.

Das	ART.Def.Nom.Sg.Neut
zu	PART.Zu
versteuernde	ADJA.Pos.Nom.Sg.Neut
Einkommen	N.Reg.Nom.Sg.Neut
sinkt	VFIN.Full.3.Sg.Pres.Ind
.	SYM.Pun.Sent

Table 1: Correct POS tags for the German sentence *Das zu versteuernde Einkommen sinkt.*

Unfortunately, the POS trigram consisting of the tags of the first three words does not occur in the Tiger corpus. (Neither does the pair consisting of the first two tags.) The unsmoothed

Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 777–784
Manchester, August 2008

context probability of the third POS tag is therefore 0. If the probability is smoothed with the backoff distribution $p(\bullet|PART.Zu)$, the most probable tag is ADJA.Pos.Acc.Sg.Fem rather than ADJA.Pos.Nom.Sg.Neut. Thus, the agreement between the article and the adjective is not checked anymore.

A closer inspection of the Tiger corpus reveals that it actually contains all the information needed to completely disambiguate each component of the POS tag ADJA.Pos.Nom.Sg.Neut:

- All words appearing after an article (ART) and the infinitive particle *zu* (PART.zu) are attributive adjectives (ADJA) (10 of 10 cases).
- All adjectives appearing after an article and a particle (PART) have the degree positive (Pos) (39 of 39 cases).
- All adjectives appearing after a nominative article and a particle have nominative case (11 of 11 cases).
- All adjectives appearing after a singular article and a particle are singular (32 of 32 cases).
- All adjectives appearing after a neuter article and a particle are neuter (4 of 4 cases).

By (1) decomposing the context probability of ADJA.Pos.Nom.Sg.Neut into a product of attribute probabilities

p(ADJA | 2:ART, 2:ART.Def, 2:ART.Nom, 2:ART.Sg, 2:ART.Neut, 1:PART, 1:PART.Zu)
∗ p(Pos| 2:ART, 2:ART.Def, 2:ART.Nom, 2:ART.Sg, 2:ART.Neut, 1:PART, 1:PART.Zu, 0:ADJA)
∗ p(Nom | 2:ART, 2:ART.Def, 2:ART.Nom, 2:ART.Sg, 2:ART.Neut, 1:PART, 1:PART.Zu, 0:ADJA, 0:ADJA.Pos)
∗ p(Sg | 2:ART, 2:ART.Def, 2:ART.Nom, 2:ART.Sg, 2:ART.Neut, 1:PART, 1:PART.Zu, 0:ADJA, 0:ADJA.Pos, 0:ADJA.Nom)
∗ p(Neut | 2:ART, 2:ART.Def, 2:ART.Nom, 2:ART.Sg, 2:ART.Neut, 1:PART, 1:PART.Zu, 0:ADJA, 0:ADJA.Pos, 0:ADJA.Nom, 0:ADJA.Sg)

and (2) selecting the relevant context attributes for the prediction of each attribute, we obtain the following expression for the context probability:

p(ADJA | ART, PART.Zu)
∗ p(Pos | 2:ART, 1:PART, 0:ADJA)
∗ p(Nom | 2:ART.Nom, 1:PART.Zu, 0:ADJA)
∗ p(Sg | 2:ART.Sg, 1:PART.Zu, 0:ADJA)
∗ p(Neut | 2:ART.Neut, 1:PART.Zu, 0:ADJA)

The conditional probability of each attribute is 1. Hence the context probability of the whole tag is also 1. Without having observed the given context, it is possible to deduce that the observed POS tag is the only possible tag in this context.

These considerations motivate an HMM tagging approach which decomposes the POS tags into a set of simple attributes, and uses decision trees to estimate the probability of each attribute. Decision trees are ideal for this task because the identification of relevant attribute combinations is at the heart of this method. The backoff smoothing methods of traditional n-gram POS taggers require an ordering of the reduced contexts which is not available, here. Discriminatively trained taggers, on the other hand, have difficulties to handle the huge number of features which are active at the same time if any possible combination of context attributes defines a separate feature.

2 Decision Trees

Decision trees (Breiman et al., 1984; Quinlan, 1993) are normally used as classifiers, i.e. they assign classes to objects which are represented as attribute vectors. The non-terminal nodes are labeled with attribute tests, the edges with the possible outcomes of a test, and the terminal nodes are labeled with classes. An object is classified by evaluating the test of the top node on the object, following the respective edge to a daughter node, evaluating the test of the daughter node, and so on until a terminal node is reached whose class is assigned to the object.

Decision Trees are turned into *probability estimation trees* by storing a probability for each possible class at the terminal nodes instead of a single result class. Figure 1 shows a probability estimation tree for the prediction of the probability of the nominative attribute of nouns.

2.1 Induction of Decision Trees

Decision trees are incrementally built by first selecting the test which splits the manually annotated training sample into the most homogeneous subsets with respect to the class. This test, which maximizes the information gain[1] wrt. the class, is

[1]The information gain measures how much the test decreases the uncertainty about the class. It is the difference between the entropy of the empirical distribution of the class variable in the training set and the weighted average entropy

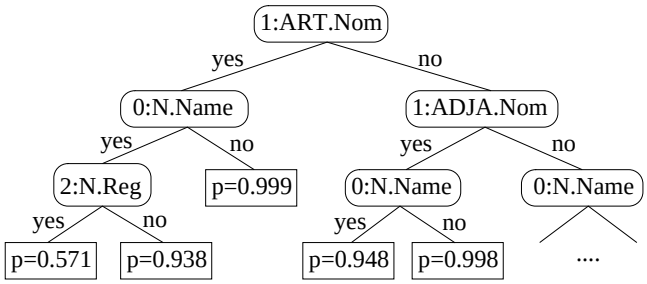

Figure 1: Probability estimation tree for the nominative case of nouns. The test *1:ART.Nom* checks if the preceding word is a nominative article.

assigned to the top node. The tree is recursively expanded by selecting the best test for each subset and so on, until all objects of the current subset belong to the same class. In a second step, the decision tree may be pruned in order to avoid overfitting to the training data.

Our tagger generates a predictor for each feature (such as base POS, number, gender etc.) Instead of using a single tree for the prediction of all possible values of a feature (such as *noun*, *article*, etc. for base POS), the tagger builds a separate decision tree for each value. The motivation was that a tree which predicts a single value (say *verb*) does not fragment the data with tests which are only relevant for the distinction of two other values (e.g. *article* and *possessive pronoun*).[2] Furthermore, we observed that such two-class decision trees require no optimization of the pruning threshold (see also section 2.2.)

The tree induction algorithm only considers binary tests, which check whether some particular attribute is present or not. The best test for each node is selected with the standard information gain criterion. The recursive tree building process terminates if the information gain is 0. The decision tree is pruned with the pruning criterion described below.

Since the tagger creates a separate tree for each attribute, the probabilities of a set of competing attributes such as *masculine*, *feminine*, and *neuter* will not exactly sum up to 1. To understand why, assume that there are three trees for the gender attributes. Two of them (say the trees for *masculine* and *feminine*) consist of a single terminal node which returns a probability of 0.3. The third tree for *neuter* has one non-terminal and two terminal nodes returning a probability of 0.3 and 0.5, respectively. The sum of probabilities is therefore either 0.9 or 1.1, but never exactly 1. This problem is solved by renormalizing the probabilities.

The probability of an attribute (such as "Nom") is always conditioned on the respective base POS (such as "N") (unless the predicted attribute is the base POS) in order to make sure that the probability of an attribute is 0 if it never appeared with the respective base POS. All context attributes other than the base POS are always used in combination with the base POS. A typical context attribute is "1:ART.Nom" which states that the preceding tag is an article with the attribute "Nom". "1:ART" is also a valid attribute specification, but "1:Nom" is not.

The tagger further restricts the set of possible test attributes by requiring that some attribute of the POS tag at position i-k (i=position of the predicted POS tag, $k \geq 1$) must have been used before an attribute of the POS tag at position i-(k+1) may be examined. This restriction improved the tagging accuracy for large contexts.

2.2 Pruning Criterion

The tagger applies[3] the critical-value pruning strategy proposed by (Mingers, 1989). A node is pruned if the information gain of the best test multiplied by the size of the data subsample is below a given threshold.

To illustrate the pruning, assume that D is the data of the current node with 50 positive and 25 negative elements, and that D_1 (with 20 positive and 20 negative elements) and D_2 (with 30 positive and 5 negative elements) are the two subsets induced by the best test. The entropy of D is $-2/3\ log_2 2/3 - 1/3\ log_2 1/3 = 0.92$, the entropy of D_1 is $-1/2\ log_2 1/2 - 1/2\ log_2 1/2 = 1$, and the entropy of D_2 is $-6/7\ log_2 6/7 - 1/7\ log_2 1/7 = 0.59$. The information gain is therefore $0.92 - (8/15 * 1 - 7/15 * 0.59) = 0.11$. The resulting score is $75 * 0.11 = 8.25$. Given a threshold of 6, the node is therefore not pruned.

We experimented with pre-pruning (where a node is always pruned if the gain is below the

in the two subsets. The weight of each subset is proportional to its size.

[2]We did not directly compare the two alternatives (two-valued vs. multi-valued tests), because the implementational effort required would have been too large.

[3]We also experimented with a pruning criterion based on binomial tests, which returned smaller trees with a slightly lower accuracy, although the difference in accuracy was never larger than 0.1% for any context size. Thus, the simpler pruning strategy presented here was chosen.

threshold) as well as post-pruning (where a node is only pruned if its sub-nodes are terminal nodes or pruned nodes). The performance of pre-pruning was slightly better and it was less dependent on the choice of the pruning threshold. A threshold of 6 consistently produced optimal or near optimal results for pre-pruning. Thus, pre-pruning with a threshold of 6 was used in the experiments.

3 Splitting of the POS Tags

The tagger treats dots in POS tag labels as attribute separators. The first attribute of a POS tag is the main category. The number of additional attributes is fixed for each main category. The additional attributes are category-specific. The *singular* attribute of a noun and an adjective POS tag are therefore two different attributes.[4]

Each position in the POS tags of a given category corresponds to a *feature*. The attributes occurring at a certain position constitute the value set of the feature.

4 Our Tagger

Our tagger is a HMM tagger which decomposes the context probabilities into a product of attribute probabilities. The probability of an attribute given the attributes of the preceding POS tags as well as the preceding attributes of the predicted POS tag is estimated with a decision tree as described before. The probabilities at the terminal nodes of the decision trees are smoothed with the parent node probabilities (which themselves were smoothed in the same way). The smoothing is implemented by adding the weighted class probabilities $p_p(c)$ of the parent node to the frequencies $f(c)$ before normalizing them to probabilities:

$$p(c) = \frac{f(c) + \alpha p_p(c)}{\alpha + \sum_c f(c)}$$

The weight α was fixed to 1 after a few experiments on development data. This smoothing strategy is closely related to Witten-Bell smoothing. The probabilities are normalized by dividing them by the total probability of all attribute values of the respective feature (see section 2.1).

The best tag sequence is computed with the Viterbi algorithm. The main differences of our tagger to a standard trigram tagger are that the order of the Markov model (the k in equation 1) is not fixed and that the context probability $p(t_i|t_{i-k}^{i-1})$ is internally computed as a product of attribute probabilities. In order to increase the speed, the tagger also applies a beam-search strategy which prunes all search paths whose probability is below the probability of the best path times a threshold. With a threshold of 10^{-3} or lower, the influence of pruning on the tagging accuracy was negligible.

4.1 Supplementary Lexicon

The tagger may use an external lexicon which supplies entries for additional words which are not found in the training corpus, and additional tags for words which did occur in the training data. If an external lexicon is provided, the lexical probabilities are smoothed as follows: The tagger computes the average tag probabilities of all words with the same set of possible POS tags. The Witten-Bell method is then applied to smooth the lexical probabilities with the average probabilities.

If the word w was observed with N different tags, and $f(w,t)$ is the joint frequency of w and POS tag t, and $p(t|[w])$ is the average probability of t among words with the same set of possible tags as w, then the smoothed probability of t given w is defined as follows:

$$p(t|w) = \frac{f(w,t) + Np(t|[w])}{f(w) + N}$$

The smoothed estimates of $p(tag|word)$ are divided by the prior probability $p(tag)$ of the tag and used instead of $p(word|tag)$.[5]

4.2 Unknown Words

The lexical probabilities of unknown words are obtained as follows: The unknown words are divided into four disjoint classes[6] with numeric expressions, words starting with an upper-case letter, words starting with a lower-case letter, and a fourth class for the other words. The tagger builds a suffix trie for each class of unknown words using the known word types from that class. The maximal length of the suffixes is 7.

The suffix tries are pruned until (i) all suffixes have a frequency of at least 5 and (ii) the information gain multiplied by the suffix frequency and di-

[4] This is the reason why the attribute tests in figure 1 used complex attributes such as *ART.Nom* rather than *Nom*.

[5] $p(word|tag)$ is equal to $p(tag|word)p(word)/p(tag)$ and $p(word)$ is a constant if the tokenization is unambiguous. Therefore dropping the factor $p(word)$ has no influence on the ranking of the different tag sequences.

[6] In earlier experiments, we had used a much larger number of word classes. Decreasing their number to 4 turned out to be better.

vided by the number of different POS tags is above a threshold of 1. More precisely, if T_α is the set of POS tags that occurred with suffix α, $|T|$ is the size of the set T, f_α is the frequency of suffix α, and $p_\alpha(t)$ is the probability of POS tag t among the words with suffix α, then the following condition must hold:

$$\frac{f_{a\alpha}}{|T_{a\alpha}|} \sum_{t \in T_{a\alpha}} p_{a\alpha}(t)\ log \frac{p_{a\alpha}(t)}{p_\alpha(t)} < 1$$

The POS probabilities are recursively smoothed with the POS probabilities of shorter suffixes using Witten-Bell smoothing.

5 Evaluation

Our tagger was first evaluated on data from the German Tiger treebank. The results were compared to those obtained with the TnT tagger (Brants, 2000) and the SVMTool (Giménez and Màrquez, 2004), which is based on support vector machines.[7] The training of the SVMTool took more than a day. Therefore it was not possible to optimize the parameters systematically. We took standard features from a 5 word window and M4-LRL training without optimization of the regularization parameter C.

In a second experiment, our tagger was also evaluated on the Czech Academic corpus 1.0 (Hladká et al., 2007) and compared to the TnT tagger.

5.1 Tiger Corpus

The German Tiger treebank (Brants et al., 2002) contains over 888,000 tokens. It is annotated with POS tags from the coarse-grained STTS tagset and with additional features encoding information about number, gender, case, person, degree, tense, and mood. After deleting problematic sentences (e.g. with an incomplete annotation) and automatically correcting some easily detectable errors, 885,707 tokens were left. The first 80% were used as training data, the first half of the rest as development data, and the last 10% as test data.

Some of the 54 STTS labels were mapped to new labels with dots, which reduced the number of main categories to 23. Examples are the nominal POS tags NN and NE which were mapped to N.Reg and N.Name. Some lexically decidable distinctions missing in the Tiger corpus have been automatically added. Examples are the distinction between definite and indefinite articles, and the distinction between hyphens, slashes, left and right parentheses, quotation marks, and other symbols which the Tiger treebank annotates with "$(".

A supplementary lexicon was created by analyzing a word list which included all words from the training, development, and test data with a German computational morphology. The analyses generated by the morphology were mapped to the Tiger tagset. Note that only the words, but not the POS tags from the test and development data were used, here. Therefore, it is always possible to create a supplementary lexicon for the corpus to be processed.

In case of the TnT tagger, the entries of the supplementary lexicon were added to the regular lexicon with a default frequency of 1 if the word/tag-pair was unknown, and with a frequency proportional to the prior probability of the tag if the word was unknown. This strategy returned the best results on the development data. In case of the SVMTool, we were not able to successfully integrate the supplementary lexicon.

5.1.1 Refined Tagset

Prepositions are not annotated with case in the Tiger treebank, although this information is important for the disambiguation of the case of the next noun phrase. In order to provide the tagger with some information about the case of prepositions, a second training corpus was created in which prepositions which always select the same case, such as *durch* (through), were annotated with this case (APPR.Acc). Prepositions which select genitive case, but also occur with dative case[8], were tagged with APPR.Gen. The more frequent ones of the remaining prepositions, such as *in* (in), were lexicalized (APPR.in). The refined tagset also distinguished between the auxiliaries *sein*, *haben*, and *werden*, and used lexicalized tags for the coordinating conjunctions *aber*, *doch*, *denn*, *wie*, *bis*, *noch*, and *als* whose distribution differs from the distribution of prototypical coordinating conjunctions such as *und* (and) or *oder* (or).

For evaluation purposes, the refined tags are mapped back to the original tags. This mapping is unambiguous.

[7] It was planned to include also the Stanford tagger (Toutanova et al., 2003) in this comparison, but it was not possible to train it on the Tiger data.

[8] In German, the genitive case of arguments is more and more replaced by the dative.

tagger	default	refined	ref.+lexicon
baseline	67.3	67.3	69.4
TnT	86.3	86.9	90.4
SVMTool	86.6	86.6	–
2 tags	87.0	87.9	91.5
10 tags	87.6	88.5	92.2

Table 2: Tagging accuracies on development data in percent. Results for 2 and for 10 preceding POS tags as context are reported for our tagger.

5.1.2 Results

Table 2 summarizes the results obtained with different taggers and tagsets on the development data. The accuracy of a baseline tagger which chooses the most probable tag[9] ignoring the context is 67.3% without and 69.4% with the supplementary lexicon.

The TnT tagger achieves 86.3% accuracy on the default tagset. A tag is considered correct if all attributes are correct. The tagset refinement increases the accuracy by about 0.6%, and the external lexicon by another 3.5%.

The SVMTool is slightly better than the TnT tagger on the default tagset, but shows little improvement from the tagset refinement. Apparently, the lexical features used by the SVMTool encode most of the information of the tagset refinement.

With a context of two preceding POS tags (similar to the trigram tagger TnT), our tagger outperforms TnT by 0.7% on the default tagset, by 1% on the refined tagset, and by 1.1% on the refined tagset plus the additional lexicon. A larger context of up to 10 preceding POS tags further increased the accuracy by 0.6, 0.6, and 0.7%, respectively.

	default	refined	ref.+lexicon
TnT STTS	97.28		
TnT Tiger	97.17	97.26	97.51
10 tags	97.39	97.57	97.97

Table 3: STTS accuracies of the TnT tagger trained on the STTS tagset, the TnT tagger trained on the Tiger tagset, and our tagger trained on the Tiger tagset.

These figures are considerably lower than e.g. the 96.7% accuracy reported in Brants (2000) for the Negra treebank which is annotated with STTS tags without agreement features. This is to be expected, however, because the STTS tagset is much smaller. Table 3 shows the results of an evaluation based on the plain STTS tagset. The first result was obtained with TnT trained on Tiger data which was mapped to STTS before. The second row contains the results for the TnT tagger when it is trained on the Tiger data and the output is mapped to STTS. The third row gives the corresponding figures for our tagger.

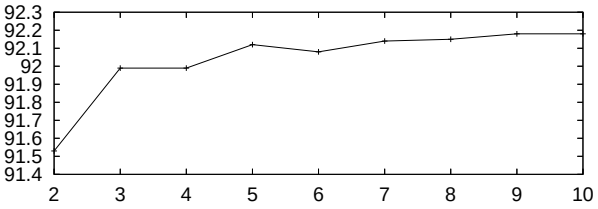

Figure 2: Tagging accuracy on development data depending on context size

Figure 2 shows that the tagging accuracy tends to increase with the context size. The best results are obtained with a context size of 10. What type of information is relevant across a distance of ten words? A good example is the decision tree for the attribute *first person* of finite verbs, which looks for a first person pronoun at positions -1 through -10 (relative to the position of the current word) in this order. Since German is a verb-final language, these tests clearly make sense.

Table 4 shows the performance on the test data. Our tagger was used with a context size of 10. The suffix length parameter of the TnT tagger was set to 6 without lexicon and to 3 with lexicon. These values were optimal on the development data. The accuracy of our tagger is lower than on the development data. This could be due to the higher rate of unknown words (10.0% vs. 7.7%). Relative to the TnT tagger, however, the accuracy is quite similar for test and development data. The differences between the two taggers are significant.[10]

tagger	default	refined	ref.+lexicon
TnT	83.45	84.11	89.14
our tagger	85.00	85.92	91.07

Table 4: Tagging accuracies on test data.

By far the most frequent tagging error was the confusion of nominative and accusative case. If

[9] Unknown words are tagged by choosing the most frequent tag of words with the same capitalization.

[10] 726 sentences were better tagged by TnT (i.e. with few errors), 1450 sentences were better tagged by our tagger. The resulting score of a binomial test is below 0.001.

this error is not counted, the tagging accuracy on the development data rises from 92.17% to 94.27%.

Our tagger is quite fast, although not as fast as the TnT tagger. With a context size of 3 (10), it annotates 7000 (2000) tokens per second on a computer with an Athlon X2 4600 CPU. The training with a context size of 10 took about 4 minutes.

5.2 Czech Academic Corpus

We also evaluated our tagger on the Czech Academic corpus (Hladká et al., 2007) which contains 652.131 tokens and about 1200 different POS tags. The data was divided into 80% training data, 10% development data and 10% test data.

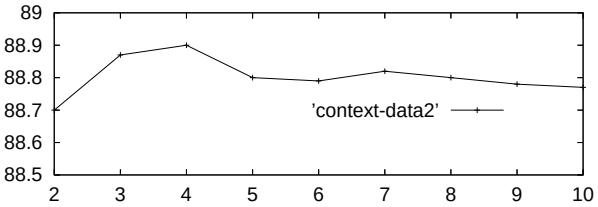

Figure 3: Accuracy on development data depending on context size

The best accuracy of our tagger on the development set was 88.9% obtained with a context of 4 preceding POS tags. The best accuracy of the TnT tagger was 88.2% with a maximal suffix length of 5. The corresponding figures for the test data are 89.53% for our tagger and 88.88% for the TnT tagger. The difference is significant.

6 Discussion

Our tagger combines two ideas, the decomposition of the probability of complex POS tags into a product of feature probabilities, and the estimation of the conditional probabilities with decision trees. A similar idea was previously presented in Kempe (1994), but apparently never applied again. The tagging accuracy reported by Kempe was below that of a traditional trigram tagger. Unlike him, we found that our tagging method out-performed state-of-the-art POS taggers on fine-grained POS tagging even if only a trigram context was used.

Schmid (1994) and Màrquez (1999) used decision trees for the estimation of contextual tag probabilities, but without a decomposition of the tag probability. Magerman (1994) applied probabilistic decision trees to parsing, but not with a generative model.

Provost & Domingos (2003) noted that well-known decision tree induction algorithms such as C4.5 (Quinlan, 1993) or CART (Breiman et al., 1984) fail to produce accurate probability estimates. They proposed to grow the decision trees to their maximal size without pruning, and to smooth the probability estimates with add-1 smoothing (also known as the Laplace correction). Ferri et al. (2003) describe a more complex backoff smoothing method. Contrary to them, we applied pruning and found that some pruning (threshold=6) gives better results than no pruning (threshold=0). Another difference is that we used N two-class trees with normalization to predict the probabilities of N classes. These two-class trees can be pruned with a fixed pruning threshold. Hence there is no need to put aside training data for parameter tuning.

An open question is whether the SVMTool (or other discriminatively trained taggers) could outperform the presented tagger if the same decomposition of POS tags and the same context size was used. We think that this might be the case if the SVM features are restricted to the set of relevant attribute combinations discovered by the decision tree, but we doubt that it is possible to train the SVMTool (or other discriminatively trained taggers) without such a restriction given the difficulties to train it with the standard context size.

Czech POS tagging has been extensively studied in the past (Hajič and Vidová-Hladká, 1998; Hajič et al., 2001; Votrubec, 2006). Spoustov et al. (2007) compared several POS taggers including an n-gram tagger and a discriminatively trained tagger (Morče), and evaluated them on the Prague Dependency Treebank (PDT 2.0). Morče's tagging accuracy was 95.12%, 0.3% better than the n-gram tagger. A hybrid system based on four different tagging methods reached an accuracy of 95.68%. Because of the different corpora used and the different amounts of lexical information available, a direct comparison to our results is difficult. Furthermore, our tagger uses no corpus-specific heuristics, whereas Morče e.g. is optimized for Czech POS tagging.

The German tagging results are, to the best of our knowledge, the first published results for fine-grained POS tagging with the Tiger tagset.

7 Summary

We presented a HMM POS tagger for fine-grained tagsets which splits the POS tags into attribute vectors and estimates the conditional probabilities of the attributes with decision trees. In experiments with German and Czech corpora, this method achieved a higher tagging accuracy than two state-of-the-art general-purpose POS taggers (TnT and SVMTool).

References

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.

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

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Wadsworth and Brooks, Pacific Grove CA.

Ferri, C., P. Flach, and J. Hernández-Orallo. 2003. Improving the AUC of probabilistic estimators trees. In *Proceedings of 14th European Conference on Machine Learning (ECML'03)*, pages 121–132.

Giménez, Jesús and Lluís Màrquez. 2004. SVMTool: A general POS tagger generator based on support vector machines. In *Proceedings of the IV International Conference on Language Resources and Evaluation (LREC'04)*, pages 43–46, Lisbon, Portugal.

Hajič, Jan and Barbora Vidová-Hladká. 1998. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of ACL-COLING'98*, Montreal, Canada.

Hajič, Jan, Pavel Krbec, Karel Oliva, Pavel Květoň, and Vladimír Petkevič. 2001. Serial combination of rules and statistics: A case study in czech tagging. In *Proceedings of the 39th Annual Meeting of the ACL*, Toulouse, France.

Hladká, Barbora Vidová, Jan Hajic, Jirí Hana, Jaroslava Hlavácová, Jirí Mírovský, and Jan Votrubec. 2007. *Czech Academic Corpus 1.0 Guide*. Karolinum Press, Prag, Czechia.

Kempe, André. 1994. Probabilistic tagging with feature structures. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 1994)*, pages 161–165, Kyoto, Japan.

Magerman, David M. 1994. *Natural Language Processing as Statistical Pattern Recognition*. Ph.D. thesis, Stanford University.

Màrquez, Lluís. 1999. *POS Tagging : A Machine Learning Approach based on Decision Trees*. Ph.D. thesis, Dep. LSI, Universitat Politecnica de Catalunya (UPC), Barcelona, Spain, July.

Mingers, John. 1989. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243.

Provost, Foster and Pedro Domingos. 2003. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215.

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

Schmid, Helmut. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK.

Spoustová, Drahomíra, Jan Hajič, Jan Votrubec, Pavel Krbec, and Pavel Květoň. 2007. The best of two worlds: Cooperation of statistical and rule-based taggers for czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing*, pages 67–74, Prague, Czech Republic, June.

Toutanova, Kristina, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259, Edmonton, Canada.

Votrubec, Jan. 2006. Morphological tagging based on averaged perceptron. In *Proceedings of the 15th Annual Conference of Doctoral Students (WDS)*, pages 191–195.