**Efficient Higher-Order CRFs for Morphological Tagging**

Thomas Mu¨ ller*†‡*, Helmut Schmid*†‡*, and Hinrich Schu¨ tze*†*

*†*Center for Information and Language Processing, University of Munich, Germany

*‡*Institute for Natural Language Processing , University of Stuttgart, Germany

[muellets@cis.lmu.de](mailto:muellets@cis.lmu.de)

**Abstract**

Training higher-order conditional random fields is prohibitive for huge tag sets. We present an approximated conditional random field using coarse-to-fine decoding and early updating. We show that our implementation yields fast and accurate morphological taggers across six languages with different morpho- logical properties and that across languages higher-order models give significant improve- ments over 1*st* -order models.

**1 Introduction**

Conditional Random Fields (CRFs) (Lafferty et al.,

2001) are arguably one of the best performing se- quence prediction models for many Natural Lan- guage Processing (NLP) tasks. During CRF train- ing forward-backward computations, a form of dy- namic programming, dominate the asymptotic run- time. The training and also decoding times thus depend polynomially on the size of the tagset and exponentially on the order of the CRF. This prob- ably explains why CRFs, despite their outstanding accuracy, normally only are applied to tasks with small tagsets such as Named Entity Recognition and Chunking; if they are applied to tasks with bigger tagsets – e.g., to part-of-speech (POS) tagging for English – then they generally are used as 1*st*-order models.

In this paper, we demonstrate that fast and accu- rate CRF training and tagging is possible for large tagsets of even thousands of tags by approximat- ing the CRF objective function using coarse-to-fine decoding (Charniak and Johnson, 2005; Rush and

Petrov, 2012). Our pruned CRF (PCRF) model has much smaller runtime than higher-order CRF mod- els and may thus lead to an even broader application of CRFs across NLP tagging tasks.

We use POS tagging and combined POS and morphological (POS+MORPH) tagging to demon- strate the properties and benefits of our approach. POS+MORPH disambiguation is an important pre- processing step for syntactic parsing. It is usually tackled by applying sequence prediction. POS+MORPH tagging is also a good example of a task where CRFs are rarely applied as the tagsets are often so big that even 1*st*-order dynamic program- ming is too expensive. A workaround is to restrict the possible tag candidates per position by using ei- ther morphological analyzers (MAs), dictionaries or heuristics (Hajicˇ, 2000). In this paper, however we show that when using pruning (i.e., PCRFs), CRFs can be trained in reasonable time, which makes hard constraints unnecessary.

In this paper, we run successful experiments on six languages with different morphological prop- erties; we interpret this as evidence that our ap- proach is a general solution to the problem of POS+MORPH tagging. The tagsets in our experi- ments range from small sizes of 12 to large sizes of up to 1811. We will see that even for the smallest tagset, PCRFs need only 40% of the training time of standard CRFs. For the bigger tagset sizes we can reduce training times from several days to several hours. We will also show that training higher-order PCRF models takes only several minutes longer than training 1*st*-order models and – depending on the language – may lead to substantial accuracy im-

322

*Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, 18-21 October 2013. *Q*c 2013 Association for Computational Linguistics

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Language | Sentences | Tokens | POS  Tags | MORPH  Tags | POS+MORPH  Tags | OOV  Rate |
| ar (Arabic)  cs (Czech) en (English) es (Spanish) de (German)  hu (Hungarian) | 15,760  38,727  38,219  14,329  40,472  61,034 | 614,050  652,544  912,344  427,442  719,530  1,116,722 | 38  12  45  12  54  57 | 516  1,811  264  255  1,028 | 516  1,811  45  303  681  1,071 | 4.58%  8.58%  3.34%  6.47%  7.64%  10.71% |

Table 1: Training set statistics. Out-Of-Vocabulary (OOV) rate is regarding the development sets.

provements. For example in German POS+MORPH tagging, a 1*st*-order model (trained in 32 minutes) achieves an accuracy of 88.96 while a 3*rd*-order model (trained in 35 minutes) achieves an accuracy

*llD* ( *λ*) =

(*""x,""y*)*∈D*

log *p*( *y| x, λ*)

of 90.60.

The remainder of the paper is structured as fol- lows: Section 2 describes our CRF implementa- tion1 and the feature set used. Section 3 sum- marizes related work on tagging with CRFs, effi- cient CRF tagging and coarse-to-fine decoding. Sec- tion 4 describes experiments on POS tagging and POS+MORPH tagging and Section 5 summarizes the main contributions of the paper.

**2 Methodology**

**2.1 Standard CRF Training**

In a standard CRF we model our sentences using a globally normalized log-linear model. The proba- bility of a tag sequence *y* given a sentence *x* is then given as:

exp *t,i λi · φi*( *y, x, t*)

*p*( *y x*) =

*|*

*Z* ( *λ, x*)

*Z* ( *λ, x*) = exp *λi · φi*( *y, x, t*)

In order to use numerical optimization we have to calculate the gradient of the log-likelihood, which is a vector of partial derivatives *∂llD* ( *λ*)*/∂λi*. For a training sentence *x, y* and a token index *t* the deriva- tive wrt feature *i* is given by:

*φi*( *y, x, t*) *−*  *φi*( *y , x, t*) *p*( *y | x, λ*)

*""y*

This is the difference between the empirical fea- ture count in the training data and the estimated count in the current model *λ*. For a 1*st*-order model, we can replace the expensive sum over all possible tag sequences *y* by a sum over all pairs of tags:

*φi*(*yt, yt*+1*, x, t*) *−*  *φi*(*y, y , x, t*) *p*(*y, y | x, λ*)

*y,y*

The probability of a tag pair *p*(*y, y | x, λ*) can then be calculated efficiently using the forward-backward algorithm. If we further reduce the complexity of the

model to a 0-order model, we obtain simple maxi-

*""y*

*t,i*

mum entropy model updates:

Where *t* and *i* are token and feature indexes, *φi* is a feature function, *λi* is a feature weight and *Z* is a normalization constant. During training the feature weights *λ* are set to maximize the conditional log- likelihood of the training data *D*:

1 Our java implementation MarMoT is available at https://code.google.com/p/cistern/

*φi*(*yt, x, t*) *−*  *φi*(*y, x, t*) *p*(*y| x, λ*)

*y*

**2.2 Pruned CRF Training**

As we discussed in the introduction, we want to de- code sentences by applying a variant of coarse-to- fine tagging. Naively, to later tag with *nth*-order

accuracy we would train a series of *n* CRFs of in- creasing order. We would then use the CRF of order

*n −* 1 to restrict the input of the CRF of order *n*.

In this paper we approximate this approach, but do

so while training only one integrated model. This way we can save both memory (by sharing feature weights between different models) and training time (by saving lower-order updates).

The main idea of our approach is to create increas- ingly complex lattices and to filter candidate states at every step to prevent a polynomial increase in lat- tice size. The first step is to create a 0-order lat- tice, which as discussed above, is identical to a se- ries of independent local maximum entropy models

*p*(*y|x, t*). The models base their prediction on the

current word *xt* and the immediate lexical context.

We then calculate the posterior probabilities and re- move states *y* with *p*(*y|x, t*) *< τ*0 from the lattice,

where *τ*0 is a parameter. The resulting reduced lat- tice is similar to what we would obtain using candi- date selection based on an MA.

We can now create a first order lattice by adding transitions to the pruned lattice and pruning with threshold *τ*1. The only difference to 0-order prun- ing is that we now have to run forward-backward

to calculate the probabilities *p*(*y|x, t*). Note that in

theory we could also apply the pruning to transition probabilities of the form *p*(*y, y |x, t*); however, this

does not seem to yield more accurate models and is less efficient than state pruning.

For higher-order lattices we merge pairs of states into new states, add transitions and prune with threshold *τi*.

We train the model using *l*1-regularized Stochas-

tic Gradient Descent (SGD) (Tsuruoka et al., 2009). We would like to create a cascade of increasingly complex lattices and update the weight vector with the gradient of the last lattice. The updates, how- ever, are undefined if the gold sequence is pruned from the lattice. A solution would be to simply rein- sert the gold sequence, but this yields poor results as the model never learns to keep the gold sequence in the lower-order lattices. As an alternative we per- form the gradient update with the highest lattice still containing the gold sequence. This approach is sim- ilar to “early updating” (Collins and Roark, 2004) in perceptron learning, where during beam search an update with the highest scoring partial hypothe-

1: **function** GETSUMLATTICE(sentence*, \_\_τ* )

2: gold-tags *←* getTags(sentence)

3: candidates *←* getAllCandidates(sentence)

4: lattice *←* ZeroOrderLattice(candidates)

5: **for** *i* = 1 *→ n* **do**

6: candidates *←* lattice*.* prune(*τi−*1 )

7: **if** gold-tags */∈* candidates **then**

8: **return** lattice

9: **end if**

10: **if** *i >* 1 **then**

11: candidates *←* mergeStates(candidates)

12: **end if**

13: candidates *←* addTransitions(candidates)

14: lattice *←* SequenceLattice(candidates*, i*)

15: **end for**

16: **return** lattice

17: **end function**

Figure 1: Lattice generation during training

sis is performed whenever the gold candidate falls out of the beam. Intuitively, we are trying to opti- mize an *nth*-order CRF objective function, but ap- ply small lower-order corrections to the weight vec- tor when necessary to keep the gold candidate in the lattice. Figure 1 illustrates the lattice generation pro- cess. The lattice generation during decoding is iden- tical, except that we always return a lattice of the highest order *n*.

The savings in training time of this integrated ap- proach are large; e.g., training a maximum entropy model over a tagset of roughly 1800 tags and more than half a million instances is slow as we have to apply 1800 weight vector updates for every token in the training set and every SGD iteration. In the integrated model we only have to apply 1800 up- dates when we lose the gold sequence during fil- tering. Thus, in our implementation training a 0- order model for Czech takes roughly twice as long as training a 1*st*-order model.

**2.3 Threshold Estimation**

Our approach would not work if we were to set the parameters *τi* to fixed predetermined values; e.g., the *τi* depend on the size of the tagset and should be adapted during training as we start the training with a uniform model that becomes more specific. We therefore set the *τi* by *specifying µi, the average number of tags per position that should remain in the lattice after pruning*. This also guarantees sta- ble lattice sizes and thus stable training times. We

achieve stable average number of tags per position by setting the *τi* dynamically during training: we measure the real average number of candidates per position *µ*ˆ*i* and apply corrections after processing a certain fraction of the sentences of the training set. The updates are of the form:

0.2

0.15

0.1

train dev

*τi* =

Unreachable gold candidates

(+0*.*1 *· τi* if *µ*ˆ*i < µi*

*−*0*.*1 *· τi* if *µ*ˆ*i > µi*

0.05

0

0 1 2 3 4 5 6 7 8 9 10

Epochs

Figure 2 shows an example training run for Ger-

man with *µ*0 = 4. Here the 0-order lattice reduces the number of tags per position from 681 to 4 losing roughly 15% of the gold sequences of the develop- ment set, which means that for 85% of the sentences the correct candidate is still in the lattice. This cor- responds to more than 99% of the tokens. We can also see that after two iterations only a very small number of 0-order updates have to be performed.

**2.4 Tag Decomposition**

As we discussed before for the very large POS+MORPH tagsets, most of the decoding time is spent on the 0-order level. To decrease the number of tag candidates in the 0-order model, we decode in two steps by separating the fully specified tag into a coarse-grained part-of-speech (POS) tag and a fine- grained MORPH tag containing the morphological features. We then first build a lattice over POS can- didates and apply our pruning strategy. In a second step we expand the remaining POS tags into all the combinations with MORPH tags that were seen in the training set. We thus build a sequence of lattices of both increasing order and increasing tag complex- ity.

**2.5 Feature Set**

We use the features of Ratnaparkhi (1996) and Man- ning (2011): the current, preceding and succeed- ing words as unigrams and bigrams and for rare words prefixes and suffixes up to length 10, and the occurrence of capital characters, digits and spe- cial characters. We define a rare word as a word

with training set frequency *≤* 10. We concate-

nate every feature with the POS and MORPH tag

and every morphological feature. E.g., for the word “der”, the POS tag art (article) and the MORPH tag gen|sg|fem (genitive, singular, feminine) we

Figure 2: Example training run of a pruned 1*st* -order model on German showing the fraction of pruned gold se- quences (= sentences) during training for training (train) and development sets (dev).

get the following features for the current word tem- plate: der+art, der+gen|sg|fem, der+gen, der+sg and der+fem.

We also use an additional binary feature, which indicates whether the current word has been seen with the current tag or – if the word is rare – whether the tag is in a set of open tag classes. The open tag classes are estimated by 10-fold cross validation on the training set: We first use the folds to estimate how often a tag is seen with an unknown word. We

then consider tags with a relative frequency *≥* 10*−*4

as open tag classes. While this is a heuristic, it is

safer to use a “soft” heuristic as a feature in the lat- tice than a hard constraint.

For some experiments we also use the output of a morphological analyzer (MA). In that case we sim- ply use every analysis of the MA as a simple nom- inal feature. This approach is attractive because it does not require the output of the MA and the an- notation of the treebank to be identical; in fact, it can even be used if treebank annotation and MA use completely different features.

Because the weight vector dimensionality is high for large tagsets and productive languages, we use a hash kernel (Shi et al., 2009) to keep the dimension- ality constant.

**3 Related Work**

Smith et al. (2005) use CRFs for POS+MORPH tag- ging, but use a morphological analyzer for candidate selection. They report training times of several days

and that they had to use simplified models for Czech.

Several methods have been proposed to reduce CRF training times. Stochastic gradient descent can be applied to reduce the training time by a factor of 5 (Tsuruoka et al., 2009) and without drastic losses in accuracy. Lavergne et al. (2010) make use of feature sparsity to significantly speed up training for mod- erate tagset sizes (*<* 100) and huge feature spaces. It is unclear if their approach would also work for huge tag sets (*>* 1000).

Coarse-to-fine decoding has been successfully ap- plied to CYK parsing where full dynamic program- ming is often intractable when big grammars are used (Charniak and Johnson, 2005). Weiss and Taskar (2010) develop cascades of models of in- creasing complexity in a framework based on per- ceptron learning and an explicit trade-off between accuracy and efficiency.

Kaji et al. (2010) propose a modified Viterbi algo- rithm that is still optimal but depending on task and especially for big tag sets might be several orders of magnitude faster. While their algorithm can be used to produce fast decoders, there is no such modifica- tion for the forward-backward algorithm used during CRF training.

**4 Experiments**

We run POS+MORPH tagging experiments on Ara- bic (ar), Czech (cs), Spanish (es), German (de) and Hungarian (hu). The following table shows the type- token (T/T) ratio, the average number of tags of ev- ery word form that occurs more than once in the training set (*A*) and the number of tags of the most ambiguous word form (*A*ˆ):

the Penn Arabic Treebank. Czech is a highly inflect- ing Slavic language with a large number of morpho- logical features. Spanish is a Romance language. Based on the statistics above we can see that it has few POS+MORPH ambiguities. It is also the lan- guage with the smallest tagset and the only language in our setup that – with a few exceptions – does not mark case. German is a Germanic language and – based on the statistics above – the language with the most ambiguous morphology. The reason is that it only has a small number of inflectional suffixes. The total number of nominal inflectional suffixes for example is five. A good example for a highly am- biguous suffix is “en”, which is a marker for infini- tive verb forms, for the 1*st* and 3*rd* person plural and for the polite 2*nd* person singular. Additionally, it marks plural nouns of all cases and singular nouns in genitive, dative and accusative case.

Hungarian is a Finno-Ugric language with an ag- glutinative morphology; this results in a high type- token ratio, but also the lowest level of word form ambiguity among the selected languages.

POS tagging experiments are run on all the lan- guages above and also on English.

**4.1 Resources**

For Arabic we use the Penn Arabic Tree- bank (Maamouri et al., 2004), parts 1–3 in their latest versions (LDC2010T08, LDC2010T13, LDC2011T09). As training set we use parts 1 and 2 and part 3 up to section ANN20020815.0083. All consecutive sections up to ANN20021015.0096 are used as development set and the remainder as test set. We use the unvocalized and pretokenized transliterations as input. For Czech and Spanish,

we use the CoNLL 2009 data sets (Hajicˇ

|  |  |  |  |
| --- | --- | --- | --- |
|  | T/T | *A* | *A*ˆ |
| ar  cs es de hu | 0.06  0.13  0.09  0.11  0.11 | 2.06  1.64  1.14  2.15  1.11 | 17  23  9  44  10 |

et al.,

Arabic is a Semitic language with nonconcate- native morphology. An additional difficulty is that vowels are often not written in Arabic script. This introduces a high number of ambiguities; on the other hand it reduces the type-token ratio, which generally makes learning easier. In this paper, we work with the transliteration of Arabic provided in

2009); for German, the TIGER treebank (Brants et al., 2002) with the split from Fraser et al. (2013); for Hungarian, the Szeged treebank (Csendes et al.,

2005) with the split from Farkas et al. (2012). For

English we use the Penn Treebank (Marcus et al.,

1993) with the split from Toutanova et al. (2003).

We also compute the possible POS+MORPH tags for every word using MAs. For Arabic we use the AraMorph reimplementation of Buckwalter (2002), for Czech the “free” morphology (Hajicˇ, 2001), for Spanish Freeling (Padro´ and Stanilovsky, 2012), for German DMOR (Schiller, 1995) and for Hungarian

Magyarlanc 2.0 (Zsibrita et al., 2013).

**4.2 Setup**

To compare the training and decoding times we run all experiments on the same test machine, which fea- tures two Hexa-Core Intel Xeon X5680 CPUs with

3,33 GHz and 6 cores each and 144 GB of mem- ory. The baseline tagger and our PCRF implemen- tation are run single threaded.2 The taggers are im- plemented in different programming languages and with different degrees of optimization; still, the run times are indicative of comparative performance to be expected in practice.

Our Java implementation is always run with 10

SGD iterations and a regularization parameter of

0*.*1, which for German was the optimal value out of

*{*0*,* 0*.*01*,* 0*.*1*,* 1*.*0*}*. We follow Tsuruoka et al. (2009)

in our implementation of SGD and shuffle the train-

ing set between epochs. All numbers shown are av- erages over 5 independent runs. Where not noted otherwise, we use *µ*0 = 4, *µ*1 = 2 and *µ*2 = 1*.*5. We found that higher values do not consistently in- crease performance on the development set, but re- sult in much higher training times.

**4.3 POS Experiments**

In a first experiment we evaluate the speed and ac- curacy of CRFs and PCRFs on the POS tagsets. As shown in Table 1 the tagset sizes range from

12 for Czech and Spanish to 54 and 57 for Ger- man and Hungarian, with Arabic (38) and English (45) in between. The results of our experiments are given in Table 2. For the 1*st*-order models, we ob- serve speed-ups in training time from 2.3 to 31 at no loss in accuracy. For all languages, training pruned higher-order models is faster than training unpruned

1*st*-order models and yields more accurate models.

Accuracy improvements range from 0.08 for Hun- garian to 0.25 for German. We can conclude that for small and medium tagset sizes PCRFs give sub- stantial improvements in both training and decod- ing speed3 and thus allow for higher-order tagging,

which for all languages leads to significant4 accu- racy improvements.

**4.4 POS+MORPH Oracle Experiments**

Ideally, for the full POS+MORPH tagset we would also compare our results to an unpruned CRF, but our implementation turned out to be too slow to do the required number of experiments. For German,

the model processed *≈* 0*.*1 sentences per second

during training; so running 10 SGD iterations on

the 40,472 sentences would take more than a month. We therefore compare our model against models that perform oracle pruning, which means we perform standard pruning, but always keep the gold candi- date in the lattice. The oracle pruning is applied dur- ing training and testing on the development set. The oracle model performance is thus an upper bound for the performance of an unpruned CRF.

The most interesting pruning step happens at the

0-order level when we reduce from hundreds of can- didates to just a couple. Table 3 shows the results for

1*st*-order CRFs.

We can roughly group the five languages into three groups: for Spanish and Hungarian the dam- age is negligible, for Arabic we see a small decrease of 0.07 and only for Czech and German we observe considerable differences of 0.14 and 0.37. Surpris- ingly, doubling the number of candidates per posi- tion does not lead to significant improvements.

We can conclude that except for Czech and Ger- man losses due to pruning are insignificant.

**4.5 POS+MORPH Higher-Order Experiments**

One argument for PCRFs is that while they might be less accurate than standard CRFs they allow to train higher-order models, which in turn might be more accurate than their standard lower-order coun- terparts. In this section, we investigate how big the improvements of higher-order models are. The re- sults are given in the following table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | ar | cs | es | de | hu |
| 1  2  3 | 90.90  91.86\*  91.88\* | 92.45  93.06\*  92.97\* | 97.95  98.01  97.87 | 88.96  90.27\*  90.60\* | 96.47  96.57\*  96.50 |

2 Our tagger might actually use more than one core because the Java garbage collection is run in parallel.

3 Decoding speeds are provided in an appendix submitted

separately.

4 Throughout the paper we establish significance by running approximate randomization tests on sentences (Yeh, 2000).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n | ar  TT ACC | cs  TT ACC | es  TT ACC | de  TT ACC | hu  TT ACC | en  TT ACC |
| CRF 1  PCRF 1  PCRF 2  PCRF 3 | 106 96.21  5 96.21  6 96.43\*  6 96.43\* | 10 98.95  4 98.96  5 99.01\*  6 99.03\* | 7 98.51  3 98.52  3 98.65\*  4 98.66\* | 234 97.69  7 97.70  9 97.91\*  9 97.94\* | 374 97.63  12 97.64  13 97.71\*  14 97.69 | 154 97.05  5 97.07  6 97.21\*  6 97.19\* |

Table 2: POS tagging experiments with pruned and unpruned CRFs with different orders *n*. For every language the training time in minutes (TT) and the POS accuracy (ACC) are given. \* indicates models significantly better than CRF (first line).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ar | cs | es | de | hu |
| 1 Oracle *µ*0 = 4  2 Model *µ*0 = 4  3 Model *µ*0 = 8 | 90.97  90.90  90.89 | 92.59  92.45\*  92.48\* | 97.91  97.95  97.94 | 89.33  88.96\*  88.94\* | 96.48  96.47  96.47 |

Table 3: Accuracies for models with and without oracle pruning. \* indicates models significantly worse than the oracle model.

We see that 2*nd*-order models give improvements for all languages. For Spanish and Hungarian we see

minor improvements *≤* 0*.*1.

For Czech we see a moderate improvement of

0.61 and for Arabic and German we observe sub- stantial improvements of 0.96 and 1.31. An analysis on the development set revealed that for all three lan- guages, case is the morphological feature that bene- fits most from higher-order models. A possible ex- planation is that case has a high correlation with syn- tactic relations and is thus affected by long-distance dependencies.

German is the only language where fourgram models give an additional improvement over trigram models. The reason seem to be sentences with long- range dependencies, e.g., “Die Rebellen haben kein Lo¨ segeld verlangt” (The rebels have not demanded any ransom); “verlangt” (demanded) is a past partic- ple that is separated from the auxilary verb “haben” (have). The 2*nd*-order model does not consider enough context and misclassifies “verlangt” as a fi- nite verb form, while the 3*rd*-order model tags it cor- rectly.

We can also conclude that the improvements for higher-order models are always higher than the loss we estimated in the oracle experiments. More pre- cisely we see that if a language has a low number of word form ambiguities (e.g., Hungarian) we observe a small loss during 0-order pruning but we also have to expect less of an improvement when increasing

the order of the model. For languages with a high number of word form ambiguities (e.g., German) we must anticipate some loss during 0-order pruning, but we also see substantial benefits for higher-order models.

Surprisingly, we found that higher-order PCRF models can also avoid the pruning errors of lower- order models. Here is an example from the German data. The word “Januar” (January) is ambiguous: in the training set, it occurs 108 times as dative, 9 times as accusative and only 5 times as nominative. The development set contains 48 nominative instances of “Januar” in datelines at the end of news articles, e.g., “TEL AVIV, 3. Januar”. For these 48 occurrences, (i) the oracle model in Table 3 selects the *correct* case nominative, (ii) the 1*st*-order PCRF model se- lects the *incorrect* case accusative, and (iii) the 2*nd*- and 3*rd*-order models select – unlike the 1*st*-order model – the *correct* case nominative. Our interpreta- tion is that the correct nominative reading is pruned from the 0-order lattice. However, the higher-order models can put less weight on 0-order features as they have access to more context to disambiguate the sequence. The lower weights of order-0 result in a more uniform posterior distribution and the nomina- tive reading is not pruned from the lattice.

**4.6 Experiments with Morph. Analyzers**

In this section we compare the improvements of higher-order models when used with MAs. The re-

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ar  TT ACC | cs  TT ACC | es  TT ACC | de  TT ACC | hu  TT ACC | en  TT ACC |
| SVMTool  Morfette CRFSuite Stanford PCRF 1  PCRF 2  PCRF 3 | 178 96.39  9 95.91  4 96.20  29 95.98  5 96.21\*  6 **96.43**  6 **96.43** | 935 98.94  6 99.00  2 99.02  8 **99.08**  4 98.96\*  5 99.01\*  6 99.03 | 64 98.42  3 98.43  2 98.40  7 98.53  3 98.52  3 98.65\*  4 **98.66\*** | 899 97.29  16 97.28  8 97.57  51 97.70  7 97.70  9 97.91\*  9 **97.94\*** | 2653 97.42  30 97.53  15 97.48  40 97.53  12 97.64\*  13 **97.71\***  14 97.69\* | 253 97.09  17 96.85  8 96.80  65 **97.24**  5 97.07\*  6 97.21  6 97.19 |

Table 4: Development results for POS tagging. Given are training times in minutes (TT) and accuracies (ACC). Best baseline results are underlined and the overall best results bold. \* indicates a significant difference (positive or negative) between the best baseline and a PCRF model.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ar | cs | es | de | hu | en |
| SVMTool  Morfette CRFSuite Stanford PCRF 1  PCRF 2  PCRF 3 | **96.19**  95.55  95.97  95.75  96.03\*  96.11  96.14 | 98.82  98.91  98.91  **98.99**  98.83\*  98.88\*  98.87\* | 98.44  98.41  98.40  98.50  98.46  **98.66\***  **98.66\*** | 96.44  96.68  96.82  97.09  97.11  97.36\*  **97.44\*** | 97.32  97.28  97.32  97.32  97.44\*  **97.50\***  97.49\* | 97.12  96.89  96.94  **97.28**  97.09\*  97.23  97.19\* |

Table 5: Test results for POS tagging. Best baseline results are underlined and the overall best results bold. \* indicates a significant difference between the best baseline and a PCRF model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ar  TT ACC | cs  TT ACC | es  TT ACC | de  TT ACC | hu  TT ACC |
| SVMTool  RFTagger Morfette CRFSuite PCRF 1  PCRF 2  PCRF 3 | 454 89.91  4 89.09  132 89.97  309 89.33  22 90.90\*  26 91.86\*  26 **91.88**\* | 2454 89.91  3 90.38  539 90.37  9274 91.10  301 92.45\*  318 **93.06**\*  318 92.97\* | 64 97.63  1 97.44  63 97.71  69 97.53  25 97.95\*  32 **98.01**\*  35 97.87\* | 1649 85.98  5 87.10  286 85.90  1295 87.78  32 88.96\*  37 90.27\*  37 **90.60**\* | 3697 95.61  10 95.06  540 95.99  5467 95.95  230 96.47\*  242 **96.57**\*  241 96.50\* |

Table 6: Development results for POS+MORPH tagging. Given are training times in minutes (TT) and accuracies (ACC). Best baseline results are underlined and the overall best results bold. \* indicates a significant difference between the best baseline and a PCRF model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ar | cs | es | de | hu |
| SVMTool  RFTagger Morfette CRFSuite PCRF 1  PCRF 2  PCRF 3 | 89.58  88.76  89.62  89.05  90.32\*  **91.29\***  91.22\* | 89.62  90.43  90.01  90.97  92.31\*  92.94\*  **92.99\*** | 97.56  97.35  97.58  97.60  97.82\*  **97.93\***  97.82\* | 83.42  84.28  83.48  85.68  86.92\*  88.48\*  **88.58\*** | 95.57  94.99  95.79  95.82  96.22\*  **96.34\***  96.29\* |

Table 7: Test results for POS+MORPH tagging. Best baseline results are underlined and the overall best results bold.

\* indicates a significant difference between the best baseline and a PCRF model.

sults are given in the following table:

n ar cs es de hu

1 90*.*90*−* 92*.*45*−* 97*.*95*−* 88*.*96*−* 96*.*47*−*

2 91*.*86+ 93*.*06 98*.*01*−* 90*.*27+ 96*.*57*−*

3 91*.*88+ 92*.*97*−* 97*.*87*−* 90*.*60+ 96*.*50*−*

MA 1 91*.*22 93*.*21 98*.*27 89*.*82 97*.*28

MA 2 92*.*16+ 93*.*87+ 98*.*37+ 91*.*31+ 97*.*51+

MA 3 92*.*14+ 93*.*88+ 98*.*28 91*.*65+ 97*.*48+

Plus and minus indicate models that are signif-

icantly better or worse than MA1. We can see that the improvements due to higher-order models are orthogonal to the improvements due to MAs for all languages. This was to be expected as MAs provide additional lexical knowledge while higher-order models provide additional information about the context. For Arabic and German the improvements of higher-order models are bigger than the improvements due to MAs.

**4.7 Comparison with Baselines**

We use the following baselines: SVMTool (Gime´nez and Ma`rquez, 2004), an SVM-based dis- criminative tagger; RFTagger (Schmid and Laws,

2008), an n-gram Hidden Markov Model (HMM) tagger developed for POS+MORPH tagging; Mor- fette (Chrupała et al., 2008), an averaged percep- tron with beam search decoder; CRFSuite (Okazaki,

2007), a fast CRF implementation; and the Stanford Tagger (Toutanova et al., 2003), a bidirectional Max- imum Entropy Markov Model. For POS+MORPH tagging, all baselines are trained on the concatena- tion of POS tag and MORPH tag. We run SVM- Tool with the standard feature set and the optimal

*c*-values *∈ {*0*.*1*,* 1*,* 10*}*. Morfette is run with the de-

fault options. For CRFSuite we use *l*2-regularized

SGD training. We use the optimal regularization pa- rameter *∈ {*0*.*01*,* 0*.*1*,* 1*.*0*}* and stop after 30 itera-

tions where we reach a relative improvement in reg- ularized likelihood of at most 0.01 for all languages. The feature set is identical to our model except for some restrictions: we only use concatenations with the full tag and we do not use the binary feature that indicates whether a word-tag combination has been observed. We also had to restrict the combinations of tag and features to those observed in the training set5. Otherwise the memory requirements would ex-

as a bidirectional 2*nd*-order model and trained us- ing OWL-BFGS. For Arabic, German and English we use the language specific feature sets and for the other languages the English feature set.

Development set results for POS tagging are shown in Table 4. We can observe that Morfette, CRFSuite and the PCRF models for different orders have training times in the same order of magnitude. For Arabic, Czech and English, the PCRF accuracy is comparable to the best baseline models. For the other languages we see improvements of 0.13 for Spanish, 0.18 for Hungarian and 0.24 for German. Evaluation on the test set confirms these results, see Table 5.6

The POS+MORPH tagging development set re- sults are presented in Table 6. Morfette is the fastest discriminative baseline tagger. In comparison with Morfette the speed up for 3*rd*-order PCRFs lies be- tween 1.7 for Czech and 5 for Arabic. Morfette gives the best baseline results for Arabic, Spanish and Hungarian and CRFSuite for Czech and Ger- man. The accuracy improvements of the best PCRF models over the best baseline models range from

0.27 for Spanish over 0.58 for Hungarian, 1.91 for Arabic, 1.96 for Czech to 2.82 for German. The test set experiments in Table 7 confirm these results.

**5 Conclusion**

We presented the pruned CRF (PCRF) model for very large tagsets. The model is based on coarse-to- fine decoding and stochastic gradient descent train- ing with early updating. We showed that for mod-

erate tagset sizes of *≈* 50, the model gives signif-

icant speed-ups over a standard CRF with negligi-

ble losses in accuracy. Furthermore, we showed that training and tagging for approximated trigram and fourgram models is still faster than standard 1*st*- order tagging, but yields significant improvements in accuracy.

In oracle experiments with POS+MORPH tagsets we demonstrated that the losses due to our approx- imation depend on the word level ambiguity of the

respective language and are moderate (*≤* 0*.*14) ex-

cept for German where we observed a loss of 0*.*37.

ceed the memory of our test machine (144 GB) for

Czech and Hungarian. The Stanford Tagger is used

5 We set the CRFSuite option possible states = 0

6 Gime´nez and Ma`rquez (2004) report an accuracy of 97.16 instead of 97.12 for SVMTool for English and Manning (2011) an accuracy of 97.29 instead of 97.28 for the Stanford tagger.

We also showed that higher order tagging – which is prohibitive for standard CRF implementations – yields significant improvements over unpruned 1*st*- order models. Analogous to the oracle experiments we observed big improvements for languages with a high level of POS+MORPH ambiguity such as Ger- man and smaller improvements for languages with less ambiguity such as Hungarian and Spanish.

**Acknowledgments**

The first author is a recipient of the Google Europe Fellowship in Natural Language Processing, and this research is supported in part by this Google Fellow- ship. This research was also funded by DFG (grant SFB 732).

**References**

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER tree- bank. In *Proceedings of the workshop on treebanks and linguistic theories*.

Tim Buckwalter. 2002. Buckwalter Arabic Morpholog- ical Analyzer Version 1.0. *Linguistic Data Consor- tium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2002L49.*

Eugene Charniak and Mark Johnson. 2005. Coarse-to- fine n-best parsing and MaxEnt discriminative rerank- ing. In *Proceedings of ACL*.

Grzegorz Chrupała, Georgiana Dinu, and Josef van Gen- abith. 2008. Learning morphology with Morfette. In *Proceedings of LREC*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*.

Michael Collins. 2002. Discriminative training meth- ods for hidden Markov models: Theory and experi- ments with perceptron algorithms. In *Proceedings of EMNLP*.

Do´ ra Csendes, Ja´nos Csirik, Tibor Gyimo´ thy, and Andra´s Kocsor. 2005. The Szeged treebank. In *Proceedings of Text, Speech and Dialogue*.

Richa´rd Farkas, Veronika Vincze, and Helmut Schmid.

2012. Dependency parsing of Hungarian: Baseline re- sults and challenges. In *Proceedings of EACL*.

Alexander Fraser, Helmut Schmid, Richa´rd Farkas, Ren- jing Wang, and Hinrich Schu¨ tze. 2013. Knowl- edge Sources for Constituent Parsing of German, a Morphologically Rich and Less-Configurational Lan- guage. *Computational Linguistics*.

Jesu´ s Gime´nez and Lluis Ma`rquez. 2004. Svmtool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of LREC*.

Jan Hajicˇ, Massimiliano Ciaramita, Richard Johans- son, Daisuke Kawahara, Maria Anto` nia Mart´ı, Llu´ıs Ma`rquez, Adam Meyers, Joakim Nivre, Sebastian Pado´ , Jan Sˇ teˇpa´nek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*.

Jan Hajicˇ. 2000. Morphological tagging: Data vs. dictio- naries. In *Proceedings of NAACL*.

Jan Hajicˇ. 2001. Czech ”Free” Morphology. *URL* [*http://ufal.mff.cuni.cz/pdt/Morphology*](http://ufal.mff.cuni.cz/pdt/Morphology) *and Tagging*. Nobuhiro Kaji, Yasuhiro Fujiwara, Naoki Yoshinaga, and

Masaru Kitsuregawa. 2010. Efficient staggered de-

coding for sequence labeling. In *Proceedings of ACL*. John Lafferty, Andrew McCallum, and Fernando Pereira.

2001. Conditional random fields: Probabilistic mod-

els for segmenting and labeling sequence data. In *Pro- ceedings of ICML*.

Thomas Lavergne, Olivier Cappe´, and Franc¸ois Yvon.

2010. Practical very large scale CRFs. In *Proceed- ings of ACL*.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic treebank: Building a large-scale annotated Arabic corpus. In *Proceedings of NEMLAR*.

Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Computational Linguistics and Intelligent Text Pro- cessing*. Springer.

Mitchell P. Marcus, Mary A. Marcinkiewicz, and Beat- rice Santorini. 1993. Building a large annotated cor- pus of English: The Penn Treebank. *Computational linguistics*.

Naoaki Okazaki. 2007. Crfsuite: A fast implemen- tation of conditional random fields (CRFs). *URL* [*http://www.chokkan.org/software/crfsuite*.](http://www.chokkan.org/software/crfsuite)

Llu´ıs Padro´ and Evgeny Stanilovsky. 2012. Freeling

3.0: Towards Wider Multilinguality. In *Proceedings of LREC*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP*.

Alexander M. Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Pro- ceedings of NAACL*.

Anne Schiller. 1995. DMOR Benutzerhandbuch. *Uni- versita¨ t Stuttgart, Institut fu¨ r maschinelle Sprachver- arbeitung*.

Helmut Schmid and Florian Laws. 2008. Estimation of conditional probabilities with decision trees and an ap- plication to fine-grained POS tagging. In *Proceedings of COLING*.

Helmut Schmid. 1994. Probabilistic part-of-speech tag- ging using decision trees. In *Proceedings of NEMLP*.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007.

Guided Learning for Bidirectional Sequence Classifi- cation. In *Proceedings of ACL*.

Qinfeng Shi, James Petterson, Gideon Dror, John Lang- ford, Alex Smola, and S.V.N. Vishwanathan. 2009. Hash Kernels for Structured Data. *J. Mach. Learn. Res.*

Noah A. Smith, David A. Smith, and Roy W. Tromble.

2005. Context-based morphological disambiguation with random fields. In *Proceedings of EMNLP*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Pro- ceedings of NAACL*.

Yoshimasa Tsuruoka, Jun’ichi Tsujii, and Sophia Ana- niadou. 2009. Stochastic gradient descent training for L1-regularized log-linear models with cumulative penalty. In *Proceedings of ACL*.

David Weiss and Ben Taskar. 2010. Structured predic- tion cascades. In *In Proceedings of AISTATS*.

Alexander Yeh. 2000. More accurate tests for the statis- tical significance of result differences. In *Proceedings of COLING*.

Ja´nos Zsibrita, Veronika Vincze, and Richa´rd Farkas.

2013. Magyarlanc 2.0: Szintaktikai elemze´s e´s fel- gyors´ıtott szo´ faji egye´rtelmu˝ s´ıte´s. In *IX. Magyar Sza´ m´ıto´ ge´pes Nyelve´szeti Konferencia*.