Efficient Higher-Order CRFs for Morphological Tagging

Thomas Müller[†‡], Helmut Schmid[†‡], and Hinrich Schütze[†]

[†]Center for Information and Language Processing, University of Munich, Germany
[‡]Institute for Natural Language Processing , University of Stuttgart, Germany
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 morphological properties and that across languages higher-order models give significant improvements over 1^{st}-order models.

1 Introduction

Conditional Random Fields (CRFs) (Lafferty et al., 2001) are arguably one of the best performing sequence prediction models for many Natural Language Processing (NLP) tasks. During CRF training forward-backward computations, a form of dynamic programming, dominate the asymptotic runtime. The training and also decoding times thus depend polynomially on the size of the tagset and exponentially on the order of the CRF. This probably 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 accurate CRF training and tagging is possible for large tagsets of even thousands of tags by approximating 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 models 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 demonstrate the properties and benefits of our approach. POS+MORPH disambiguation is an important preprocessing 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 programming is too expensive. A workaround is to restrict the possible tag candidates per position by using either morphological analyzers (MAs), dictionaries or heuristics (Hajič, 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 properties; we interpret this as evidence that our approach is a general solution to the problem of POS+MORPH tagging. The tagsets in our experiments 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-

Language	Sentences	Tokens	POS Tags	MORPH Tags	POS+MORPH Tags	OOV Rate
ar (Arabic)	15,760	614,050	38	516	516	4.58%
cs (Czech)	38,727	652,544	12	1,811	1,811	8.58%
en (English)	38,219	912,344	45		45	3.34%
es (Spanish)	14,329	427,442	12	264	303	6.47%
de (German)	40,472	719,530	54	255	681	7.64%
hu (Hungarian)	61,034	1,116,722	57	1,028	1,071	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 of 90.60.

The remainder of the paper is structured as follows: Section 2 describes our CRF implementation[1] and the feature set used. Section 3 summarizes related work on tagging with CRFs, efficient CRF tagging and coarse-to-fine decoding. Section 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 probability of a tag sequence $\vec{y}$ given a sentence $\vec{x}$ is then given as:

$$p(\vec{y}|\vec{x}) = \frac{\exp \sum_{t,i} \lambda_i \cdot \phi_i(\vec{y}, \vec{x}, t)}{Z(\vec{\lambda}, \vec{x})}$$
$$Z(\vec{\lambda}, \vec{x}) = \sum_{\vec{y}} \exp \sum_{t,i} \lambda_i \cdot \phi_i(\vec{y}, \vec{x}, t)$$

Where t and i are token and feature indexes, ϕ_i is a feature function, λ_i 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/

$$ll_D(\vec{\lambda}) = \sum_{(\vec{x},\vec{y}) \in D} \log\, p(\vec{y}|\vec{x}, \vec{\lambda})$$

In order to use numerical optimization we have to calculate the gradient of the log-likelihood, which is a vector of partial derivatives $\partial ll_D(\vec{\lambda})/\partial\lambda_i$. For a training sentence $\vec{x}, \vec{y}$ and a token index t the derivative wrt feature i is given by:

$$\phi_i(\vec{y}, \vec{x}, t) - \sum_{\vec{y}'} \phi_i(\vec{y}', \vec{x}, t)\, p(\vec{y}'|\vec{x}, \vec{\lambda})$$

This is the difference between the empirical feature count in the training data and the estimated count in the current model $\vec{\lambda}$. For a 1^{st}-order model, we can replace the expensive sum over all possible tag sequences $\vec{y}'$ by a sum over all pairs of tags:

$$\phi_i(y_t, y_{t+1}, \vec{x}, t) - \sum_{y,y'} \phi_i(y, y', \vec{x}, t)\, p(y, y'|\vec{x}, \vec{\lambda})$$

The probability of a tag pair $p(y, y'|\vec{x}, \vec{\lambda})$ 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 maximum entropy model updates:

$$\phi_i(y_t, \vec{x}, t) - \sum_{y} \phi_i(y, \vec{x}, t)\, p(y|\vec{x}, \vec{\lambda})$$

2.2 Pruned CRF Training

As we discussed in the introduction, we want to decode sentences by applying a variant of coarse-to-fine tagging. Naively, to later tag with n^{th}-order accuracy we would train a series of n CRFs of increasing 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 increasingly complex lattices and to filter candidate states at every step to prevent a polynomial increase in lattice size. The first step is to create a 0-order lattice, which as discussed above, is identical to a series of independent local maximum entropy models $p(y|\vec{x}, t)$. The models base their prediction on the current word x_t and the immediate lexical context. We then calculate the posterior probabilities and remove states y with $p(y|\vec{x}, t) < \tau_0$ from the lattice, where τ_0 is a parameter. The resulting reduced lattice is similar to what we would obtain using candidate 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 pruning is that we now have to run forward-backward to calculate the probabilities $p(y|\vec{x}, t)$. Note that in theory we could also apply the pruning to transition probabilities of the form $p(y, y'|\vec{x}, t)$; 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 τ_i.

We train the model using l_1-regularized Stochastic 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, however, are undefined if the gold sequence is pruned from the lattice. A solution would be to simply reinsert 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 perform the gradient update with the highest lattice still containing the gold sequence. This approach is similar to "early updating" (Collins and Roark, 2004) in perceptron learning, where during beam search an update with the highest scoring partial hypothesis is performed whenever the gold candidate falls out of the beam. Intuitively, we are trying to optimize an n^{th}-order CRF objective function, but apply small lower-order corrections to the weight vector when necessary to keep the gold candidate in the lattice. Figure 1 illustrates the lattice generation process. The lattice generation during decoding is identical, except that we always return a lattice of the highest order n.

```
1: function GetSumLattice(sentence, $\vec{\tau}$)
2:     gold-tags ← getTags(sentence)
3:     candidates ← getAllCandidates(sentence)
4:     lattice ← ZeroOrderLattice(candidates)
5:     for $i = 1 \rightarrow n$ do
6:         candidates ← lattice. prune($\tau_{i-1}$)
7:         if gold-tags $\not\in$ 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

The savings in training time of this integrated approach 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 updates when we lose the gold sequence during filtering. Thus, in our implementation training a 0-order model for Czech takes roughly twice as long as training a 1st-order model.

2.3 Threshold Estimation

Our approach would not work if we were to set the parameters τ_i to fixed predetermined values; e.g., the τ_i 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 τ_i by *specifying μ_i, the average number of tags per position that should remain in the lattice after pruning*. This also guarantees stable lattice sizes and thus stable training times. We achieve stable average number of tags per position by setting the τ_i dynamically during training: we measure the real average number of candidates per position $\hat{\mu}_i$ and apply corrections after processing a certain fraction of the sentences of the training set. The updates are of the form:

$$\tau_i = \begin{cases} +0.1 \cdot \tau_i & \text{if } \hat{\mu}_i < \mu_i \\ -0.1 \cdot \tau_i & \text{if } \hat{\mu}_i > \mu_i \end{cases}$$

Figure 2 shows an example training run for German with $\mu_0 = 4$. 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 development set, which means that for 85% of the sentences the correct candidate is still in the lattice. This corresponds 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 candidates 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 complexity.

2.5 Feature Set

We use the features of Ratnaparkhi (1996) and Manning (2011): the current, preceding and succeeding words as unigrams and bigrams and for rare words prefixes and suffixes up to length 10, and the occurrence of capital characters, digits and special characters. We define a rare word as a word with training set frequency $\leq$ 10. We concatenate 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

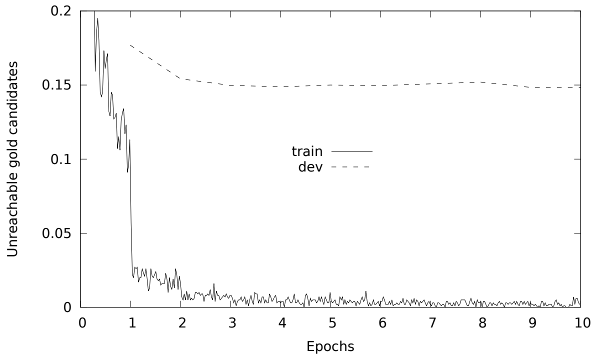

Figure 2: Example training run of a pruned 1^{st}-order model on German showing the fraction of pruned gold sequences (= sentences) during training for training (train) and development sets (dev).

get the following features for the current word template: `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 $\geq 10^{-4}$ as open tag classes. While this is a heuristic, it is safer to use a “soft” heuristic as a feature in the lattice than a hard constraint.

For some experiments we also use the output of a morphological analyzer (MA). In that case we simply use every analysis of the MA as a simple nominal feature. This approach is attractive because it does not require the output of the MA and the annotation 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 dimensionality constant.

3 Related Work

Smith et al. (2005) use CRFs for POS+MORPH tagging, 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 moderate 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 applied to CYK parsing where full dynamic programming is often intractable when big grammars are used (Charniak and Johnson, 2005). Weiss and Taskar (2010) develop cascades of models of increasing complexity in a framework based on perceptron learning and an explicit trade-off between accuracy and efficiency.

Kaji et al. (2010) propose a modified Viterbi algorithm 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 modification for the forward-backward algorithm used during CRF training.

4 Experiments

We run POS+MORPH tagging experiments on Arabic (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 every word form that occurs more than once in the training set ($\overline{A}$) and the number of tags of the most ambiguous word form ($\hat{A}$):

	T/T	$\overline{A}$	$\hat{A}$
ar	0.06	2.06	17
cs	0.13	1.64	23
es	0.09	1.14	9
de	0.11	2.15	44
hu	0.11	1.11	10

Arabic is a Semitic language with nonconcatenative 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 the Penn Arabic Treebank. Czech is a highly inflecting Slavic language with a large number of morphological 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 language 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 ambiguous suffix is "en", which is a marker for infinitive 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 agglutinative 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 languages above and also on English.

4.1 Resources

For Arabic we use the Penn Arabic Treebank (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 (Hajič et al., 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 (Hajič, 2001), for Spanish Freeling (Padró 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 features two Hexa-Core Intel Xeon X5680 CPUs with 3,33 GHz and 6 cores each and 144 GB of memory. The baseline tagger and our PCRF implementation are run single threaded.[2] The taggers are implemented 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 training set between epochs. All numbers shown are averages over 5 independent runs. Where not noted otherwise, we use $\mu_0 = 4$, $\mu_1 = 2$ and $\mu_2 = 1.5$. We found that higher values do not consistently increase performance on the development set, but result in much higher training times.

4.3 POS Experiments

In a first experiment we evaluate the speed and accuracy 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 German 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 observe 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 Hungarian to 0.25 for German. We can conclude that for small and medium tagset sizes PCRFs give substantial improvements in both training and decoding speed[3] and thus allow for higher-order tagging, which for all languages leads to significant[4] accuracy 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 candidate in the lattice. The oracle pruning is applied during 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 candidates 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 damage 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. Surprisingly, doubling the number of candidates per position does not lead to significant improvements.

We can conclude that except for Czech and German 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 counterparts. In this section, we investigate how big the improvements of higher-order models are. The results 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

[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).

		ar		cs		es		de		hu		en	
	n	TT	ACC	TT	ACC	TT	ACC	TT	ACC	TT	ACC	TT	ACC
CRF	1	106	96.21	10	98.95	7	98.51	234	97.69	374	97.63	154	97.05
PCRF	1	5	96.21	4	98.96	3	98.52	7	97.70	12	97.64	5	97.07
PCRF	2	6	96.43*	5	99.01*	3	98.65*	9	97.91*	13	97.71*	6	97.21*
PCRF	3	6	96.43*	6	99.03*	4	98.66*	9	97.94*	14	97.69	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 $\mu_0 = 4$	90.97	92.59	97.91	89.33	96.48
2	Model $\mu_0 = 4$	90.90	92.45*	97.95	88.96*	96.47
3	Model $\mu_0 = 8$	90.89	92.48*	97.94	88.94*	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 substantial improvements of 0.96 and 1.31. An analysis on the development set revealed that for all three languages, case is the morphological feature that benefits most from higher-order models. A possible explanation is that case has a high correlation with syntactic 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 Lösegeld verlangt” (The rebels have not demanded any ransom); “verlangt” (demanded) is a past participle that is separated from the auxilary verb “haben” (have). The 2^{nd}-order model does not consider enough context and misclassifies “verlangt” as a finite verb form, while the 3^{rd}-order model tags it correctly.

We can also conclude that the improvements for higher-order models are always higher than the loss we estimated in the oracle experiments. More precisely 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 selects 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 interpretation 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 nominative 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		cs		es		de		hu		en	
	TT	ACC	TT	ACC	TT	ACC	TT	ACC	TT	ACC	TT	ACC
SVMTool	178	96.39	935	98.94	64	98.42	899	97.29	2653	97.42	253	97.09
Morfette	9	95.91	6	99.00	3	98.43	16	97.28	30	97.53	17	96.85
CRFSuite	4	96.20	2	99.02	2	98.40	8	97.57	15	97.48	8	96.80
Stanford	29	95.98	8	**99.08**	7	98.53	51	97.70	40	97.53	65	**97.24**
PCRF 1	5	96.21*	4	98.96*	3	98.52	7	97.70	12	97.64*	5	97.07*
PCRF 2	6	**96.43**	5	99.01*	3	98.65*	9	97.91*	13	**97.71***	6	97.21
PCRF 3	6	**96.43**	6	99.03	4	**98.66***	9	**97.94***	14	97.69*	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	**96.19**	98.82	98.44	96.44	97.32	97.12
Morfette	95.55	98.91	98.41	96.68	97.28	96.89
CRFSuite	95.97	98.91	98.40	96.82	97.32	96.94
Stanford	95.75	**98.99**	98.50	97.09	97.32	**97.28**
PCRF 1	96.03*	98.83*	98.46	97.11	97.44*	97.09*
PCRF 2	96.11	98.88*	**98.66***	97.36*	**97.50***	97.23
PCRF 3	96.14	98.87*	**98.66***	**97.44***	97.49*	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		cs		es		de		hu	
	TT	ACC	TT	ACC	TT	ACC	TT	ACC	TT	ACC
SVMTool	454	89.91	2454	89.91	64	97.63	1649	85.98	3697	95.61
RFTagger	4	89.09	3	90.38	1	97.44	5	87.10	10	95.06
Morfette	132	89.97	539	90.37	63	97.71	286	85.90	540	95.99
CRFSuite	309	89.33	9274	91.10	69	97.53	1295	87.78	5467	95.95
PCRF 1	22	90.90*	301	92.45*	25	97.95*	32	88.96*	230	96.47*
PCRF 2	26	91.86*	318	**93.06***	32	**98.01***	37	90.27*	242	**96.57***
PCRF 3	26	**91.88***	318	92.97*	35	97.87*	37	**90.60***	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	89.58	89.62	97.56	83.42	95.57
RFTagger	88.76	90.43	97.35	84.28	94.99
Morfette	89.62	90.01	97.58	83.48	95.79
CRFSuite	89.05	90.97	97.60	85.68	95.82
PCRF 1	90.32*	92.31*	97.82*	86.92*	96.22*
PCRF 2	**91.29***	92.94*	**97.93***	88.48*	**96.34***
PCRF 3	91.22*	**92.99***	97.82*	**88.58***	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 significantly 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 (Giménez and Màrquez, 2004), an SVM-based discriminative tagger; RFTagger (Schmid and Laws, 2008), an n-gram Hidden Markov Model (HMM) tagger developed for POS+MORPH tagging; Morfette (Chrupała et al., 2008), an averaged perceptron with beam search decoder; CRFSuite (Okazaki, 2007), a fast CRF implementation; and the Stanford Tagger (Toutanova et al., 2003), a bidirectional Maximum Entropy Markov Model. For POS+MORPH tagging, all baselines are trained on the concatenation of POS tag and MORPH tag. We run SVMTool with the standard feature set and the optimal c-values $\in \{0.1, 1, 10\}$. Morfette is run with the default options. For CRFSuite we use l_2-regularized SGD training. We use the optimal regularization parameter $\in \{0.01, 0.1, 1.0\}$ and stop after 30 iterations where we reach a relative improvement in regularized 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 set[5]. Otherwise the memory requirements would exceed the memory of our test machine (144 GB) for Czech and Hungarian. The Stanford Tagger is used as a bidirectional 2^{nd}-order model and trained using 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 results 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 between 1.7 for Czech and 5 for Arabic. Morfette gives the best baseline results for Arabic, Spanish and Hungarian and CRFSuite for Czech and German. 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 training with early updating. We showed that for moderate tagset sizes of ≈ 50, the model gives significant speed-ups over a standard CRF with negligible 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 approximation depend on the word level ambiguity of the respective language and are moderate (≤ 0.14) except for German where we observed a loss of 0.37.

[5]We set the CRFSuite option $\mathrm{possible\_states} = 0$

[6]Giménez and Màrquez (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 German 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 Fellowship. 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 treebank. In *Proceedings of the workshop on treebanks and linguistic theories*.

Tim Buckwalter. 2002. Buckwalter Arabic Morphological Analyzer Version 1.0. *Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2002L49*.

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

Grzegorz Chrupała, Georgiana Dinu, and Josef van Genabith. 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 methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Dóra Csendes, János Csirik, Tibor Gyimóthy, and András Kocsor. 2005. The Szeged treebank. In *Proceedings of Text, Speech and Dialogue*.

Richárd Farkas, Veronika Vincze, and Helmut Schmid. 2012. Dependency parsing of Hungarian: Baseline results and challenges. In *Proceedings of EACL*.

Alexander Fraser, Helmut Schmid, Richárd Farkas, Renjing Wang, and Hinrich Schütze. 2013. Knowledge Sources for Constituent Parsing of German, a Morphologically Rich and Less-Configurational Language. *Computational Linguistics*.

Jesús Giménez and Lluis Màrquez. 2004. Svmtool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of LREC*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *Proceedings of NAACL*.

Jan Hajič. 2001. Czech ”Free” Morphology. *URL http://ufal.mff.cuni.cz/pdt/Morphology_and_Tagging*.

Nobuhiro Kaji, Yasuhiro Fujiwara, Naoki Yoshinaga, and Masaru Kitsuregawa. 2010. Efficient staggered decoding for sequence labeling. In *Proceedings of ACL*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings 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 Processing*. Springer.

Mitchell P. Marcus, Mary A. Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*.

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

Lluís Padró 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 *Proceedings of NAACL*.

Anne Schiller. 1995. DMOR Benutzerhandbuch. *Universität Stuttgart, Institut für maschinelle Sprachverarbeitung*.

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

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

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided Learning for Bidirectional Sequence Classification. In *Proceedings of ACL*.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, 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 *Proceedings of NAACL*.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 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 prediction cascades. In *In Proceedings of AISTATS*.

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

János Zsibrita, Veronika Vincze, and Richárd Farkas. 2013. Magyarlanc 2.0: Szintaktikai elemzés és felgyorsított szófaji egyértelműsítés. In *IX. Magyar Számítógépes Nyelvészeti Konferencia*.