Learning the Structure of Variable-Order CRFs: a Finite-State Perspective

Thomas Lavergne and **François Yvon**
LIMSI, CNRS, Univ. Paris-Sud, Université Paris Saclay
Campus Universitaire, F-91 403 Orsay, France
{lavergne,yvon}@limsi.fr

Abstract

The computational complexity of linear-chain Conditional Random Fields (CRFs) makes it difficult to deal with very large label sets and long range dependencies. Such situations are not rare and arise when dealing with morphologically rich languages or joint labelling tasks. We extend here recent proposals to consider variable order CRFs. Using an effective finite-state representation of variable-length dependencies, we propose new ways to perform feature selection at large scale and report experimental results where we outperform strong baselines on a tagging task.

1 Introduction

Conditional Random Fields (CRFs) (Lafferty et al., 2001; Sutton and McCallum, 2006) are a method of choice for many sequence labelling tasks such as Part of Speech (PoS) tagging, Text Chunking, or Named Entity Recognition. Linear-chain CRFs are easy to train by solving a convex optimization problem, can accomodate rich feature patterns, and enjoy polynomial exact inference procedures. They also deliver state-of-the-art performance for many tasks, sometimes surpassing seq2seq neural models (Schnober et al., 2016).

A major issue with CRFs is the complexity of training and inference procedures, which are quadratic in the number of possible output labels for first order models and grow exponentially when higher order dependencies are considered. This is problematic for tasks such as precise PoS tagging for Morphologically Rich Languages (MRLs), where the number of morphosyntactic labels is in the thousands (Hajič, 2000; Müller et al., 2013). Large label sets also naturally arise when *joint labelling tasks* (eg. simultaneous PoS tagging and text chunking) are considered, For such tasks, processing first-order models is demanding, and full size higher-order models are out of the question. Attempts to overcome this difficulty are based on a greedy approach which starts with first-order dependencies between labels and iteratively increases the scope of dependency patterns under the constraint that a high-order dependency is selected only if it extends an existing lower order feature (Müller et al., 2013). As a result, feature selection may only choose only few higher-order features, motivating the need for an effective variable-order CRF (voCRF) training procedure (Ye et al., 2009).[1] The latest implementation of this idea (Vieira et al., 2016) relies on (structured) sparsity promoting regularization (Martins et al., 2011) and on finite-state techniques, handling high-order features at a small extra cost (see § 2). In this approach, the sparse set of label dependency patterns is represented in a finite-state automaton, which arises as the result of the feature selection process.

In this paper, we somehow reverse the perspective and consider VoCRF training mostly as an automaton inference problem. This leads us to consider alternative techniques for learning the finite-state machine representing the dependency structure of sparse VoCRFs (see § 3). Two lines of enquiries are explored: (a) to take into account the internal structure of large tag sets in order to learn better and/or leaner feature sets; (b) to detect unconditional structural dependencies in label sequences in order to speed-up the discovery of useful features. These ideas are implemented in 6 feature selection strategies, allowing us to explore a large set of dependency structures. Relying on lazy finite-state operations, we train VoCRFs up to order 5, and achieve PoS tagging performance that

[1]This is reminiscent of *variable order HMMs*, introduced eg. in (Schütze and Singer, 1994; Ron et al., 1996).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 433–439
Copenhagen, Denmark, September 7–11, 2017.

surpass strong baselines for two MRLs (see § 4).

2 Variable order CRFs

In this section, we recall the basics of CRFs and VoCRFs and introduce some notations.

2.1 Basics

First-order CRFs use the following model:

$$p_\theta(\mathbf{y}|\mathbf{x}) = Z_\theta(\mathbf{x})^{-1} \exp(\theta^T F(\mathbf{x},\mathbf{y})) \quad (1)$$

where $\mathbf{x} = (x_1, \ldots, x_T)$ and $\mathbf{y} = (y_1, \ldots, y_T)$ are the input (in $\mathcal{X}^T$) and output (in $\mathcal{Y}^T$) sequences and $Z_\theta(\mathbf{x})$ is a normalizer. Each component $F_j(\mathbf{x}, \mathbf{y})$ of the global feature vector decomposes as a sum of local features $\sum_{t=1}^{T} f_j(y_{t-1}, y_t, x_t)$ and is associated to parameter θ_j. Local features typically use binary tests and take the form:

$$f_{u,g}(y_{t-1}, y_t, x, t) = \mathbb{I}(y_t = u \wedge g(x,t))$$
$$f_{uv,g}(y_{t-1}, y_t, x, t) = \mathbb{I}(y_{t-1}y_t = uv \wedge g(x,t))$$

where $\mathbb{I}()$ is an indicator function and $g()$ tests a local property of $\mathbf{x}$ around x_t. In this setting, the number of parameters is $|\mathcal{Y}|^2 \times |\mathcal{X}|_{\text{train}}$, where $|A|$ is the cardinality of A and $|\mathcal{X}|_{\text{train}}$ is the number of values of $g(x,t)$ observed in the training set. Even in moderate size applications, the parameter set can be very large and contain dozen of millions of features, due to the introduction of sequential dependencies in the model.

Given N i.i.d. sequences $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^N$, estimation is based on the minimization of the negated conditional log-likelihood $l(\theta)$. Optimizing this objective requires to compute its gradient and to repeatedly evaluate the conditional expectation of the feature vector. This can be done using a forward-backward algorithm having a complexity that grows quadratically with $|\mathcal{Y}|$. $l(\theta)$ is usually complemented with a regularization term so as to avoid overfitting and stabilize the optimization. Common regularizers use the ℓ_1- or the ℓ_2-norm of the parameter vector, the former having the benefit to promote sparsity, thereby performing automatic feature selection (Tibshirani, 1996).

2.2 Variable order CRFs (VoCRFs)

When the label set is large, many pairs of labels never occur in the training data and the sparsity of label ngrams quickly increases with the order p of the model. In the *variable order CRF* model, it is assumed that only a small number of ngrams (out of $|\mathcal{Y}|^p$) are associated with a non-zero parameter value. Denoting $\mathcal{W}$ the set of such ngrams and $w \in \mathcal{W}$, a generic feature function is then $f_{w,g}(w, x, t) = \mathbb{I}(y_{t-s} \ldots y_t = w \wedge g(x,t))$.

Algorithm 1: Building $A[\mathcal{W}]$

$\mathcal{W}$: list of patterns, $A[\mathcal{W}]$ initially empty
$\mathcal{U} = \texttt{Pref}(\mathcal{W})$
foreach $w \in \mathcal{W}$ **do**
  `TrieInsert`$(w, A[\mathcal{W}])$
// Add missing transitions
foreach $u = vy \in \mathcal{U}$ **do**
  `new FailureTrans`$(u, \texttt{LgSuff}(v,\mathcal{U}))$

In (order-p) VoCRFs, the computational cost of training and inference is proportional to the size of a finite-state automaton $A[\mathcal{W}]$ encoding the patterns in $\mathcal{W}$,[2] which can be much less than $|\mathcal{Y}|^p$. Our procedure for building $A[\mathcal{W}]$ is sketched in Algorithm 1, where `TrieInsert` inserts a string in a trie, `Pref`$(\mathcal{W})$ computes the set of prefixes of the strings in $\mathcal{W}$,[3] `LgSuff`$(v,\mathcal{U})$ returns the longest suffix of v in $\mathcal{U}$, and `FailureTrans` is a special ε-transition used only when no labelled transition exists (Allauzen et al., 2003).[4] Each state (or pattern prefix) v in $A[\mathcal{W}]$ is associated with a set of feature functions $\{f_{u,g}, \forall u \in \texttt{Suff}(v), g\}$.[5] The forward step of the gradient computation maintains one value $\alpha(v,t)$ per state and time step, which is recursively accumulated over all paths ending in v at time t.

The next question is to identify $\mathcal{W}$. The simplest method keeps all the ngrams viewed in training, additionally filtering rare patterns (Cuong et al., 2014). However, frequency based feature selection does not take interactions into account and is not the best solution. Ideally, one would like to train a complete order-p model with a sparsity promoting penalty, a technique that only works for small label sets.[6] The greedy algorithm of

[2]More precisely, Vieira et al. (2016) consider $\overline{\mathcal{W}}$, the closure of $\mathcal{W}$ under suffix and last character substitution, which factors as $\overline{\mathcal{W}} = \mathcal{H} \times \mathcal{Y}$. The complexity of training depends on the size of the finite-state automaton representing $\overline{\mathcal{W}}$.

[3]A trie has one state for each prefix.

[4]This was also suggested by Cotterell and Eisner (2015) as a way to build a more compact pattern automaton.

[5]Upon reaching a state v, we need to access the features that fire for that pattern, and also for all its suffixes. Each state thus stores a set of pattern; each pattern is associated with a set of tests on the observation (cf. 2.1).

[6]Recall that the size of parameter set is exponential wrt. the model order.

Schmidt and Murphy (2010); Vieira et al. (2016) is more scalable: it starts with all unigram patterns and iteratively grows $\mathcal{W}$ by extending the ngrams that have been selected in the simpler model. At each round of training, feature selection is performed using a ℓ_1 penalty and identifies the patterns that will be further augmented.

3 Learning patterns

We introduce now several alternatives for learning $\mathcal{W}$. Our motivation for doing so is twofold: (a) to take the internal structure of large label sets into account; (b) to identify more abstract patterns in label sequences, possibly containing gaps or iterations, which could yield smaller $A[\mathcal{W}]$. As discussed below, both motivations can be combined.

3.1 Greedy ℓ_1

The greedy strategy iteratively grows patterns up to order p. Considering all possible unigram and bigram patterns, we train a sparse model to select a first set of useful bigrams. In subsequent iterations, each pattern w selected at order k is extended in all possible ways to specify the pattern set at order $k+1$, which will be filtered during the next training round. This approach is close, yet simpler, than the group lasso approach of Vieira et al. (2016) and experimentally yields slightly smaller pattern sets (see Table 2). This is because we do not enforce closure under last-character replacement: once pattern w is pruned, longer patterns ending in w are never considered.[7]

3.2 Component-wise training

Large tag sets often occur in joint tasks, where multiple levels of information are encoded in one compound tag. For instance, the fine grain labels in the Tiger corpus (Brants et al., 2002) combine PoS and morphological information in tags such as `NN.Dat.Sg.Fem` for a feminine singular dative noun. In the sequel, we refer to each piece of information as a tag *component*. We assume that all tags contain the same components, using a "non-applicable" value whenever needed. Using features that test arbitrary combinations of tag components would make feature selection much more difficult, as the number of possible patterns grows combinatorially with the number of components. We keep things simple by allowing features to only evaluate one single component at a time: this allows us to identify dependencies of different orders for each component.

Assuming that each tag y contains K components $y = [z_1, z_2 \ldots, z_K]$, with $z_k \in \mathcal{Y}_k$, $\mathcal{W}$ is then computed as in § 3.1, except that we now consider one distinct set of patterns $\mathcal{W}_k$ for each component k. At each training round, each set $\mathcal{W}_k$ is extended and pruned independently from the others. Note that all these automata are trained simultaneously using a common set of features. This process results in K automata, which are intersected on the fly[8] using "lazy" composition. In our experiments, we also consider the case where we additionally combine the automaton representing complete tag sequences: this has the beneficial effect to restrict the combinations of subtags to values that actually exist in the data.

3.3 Pruned language models

Another approach for computing $\mathcal{W}$ assumes that useful dependencies between tags can be identified using an auxiliary language model (LM) trained *without paying any attention to observation sequences*. A pattern w will then be deemed useful for the labelling task only if w is a useful history in a LM of tag sequences. This strategy was implemented by first training a compact p-gram LM with entropy pruning[9] (Stolcke, 1998) and including all the surviving histories in $\mathcal{W}$. In a second step, we train the complete CRF as usual, with all observation features and ℓ_1 penalty to further prune the parameter set.

	cz	de
train set	38,727	40,474
development set	5,228	5,000
test set	4,213	5,000
# PoS	12	54
# attributes	13	8
# full tags	1,924	781

Table 1: Corpus description

3.4 Maximum entropy language models

Another technique, which combines the two previous ideas, relies on Maximum Entropy LMs

[7]cf. the discussion in (Vieira et al., 2016, § 4).

[8]Formally, each $A[\mathcal{W}_k]$ has transitions labelled with elements of $\mathcal{Y}_k$; lazy intersection operates on "generalized" transitions, where each label z is replaced with $[?, \ldots, z, \ldots, ?]$, where ? matches any symbol. $A[\mathcal{W}]$ is the intersection $\bigcap_k A[\mathcal{W}_k]$ and is labelled with completely specified tags.

[9]Starting with a full back-off n-gram language model, this approach discards n-grams if their removal causes a sufficiently small drop in cross-entropy. We used the implementation of Stolcke (2002).

(MELMs) (Rosenfeld, 1996). MELMs decompose the probabililty of a sequence $y_1 \dots y_T$ using the chain rule, where each term $p_\lambda(y_t|y_{<t})$ is a locally normalized exponential model including all possible ngram features up to order p:

$$p(y_t|y_{<t};\lambda) = Z(\lambda)^{-1} \exp \lambda^T G(y_1 \dots y_t)$$

In contrast to globally normalized models, the complexity of training remains linear wrt. $|\mathcal{Y}|$, irrespective of p. It it also straightforward both to (a) use a ℓ_1 penalty to perform feature selection; (b) include features that only test specific components of a complex tag. For an order p model, our feature functions evaluate all n-grams (for $n \leq p$) of complete tags or of one specific component:

$$\begin{aligned} G_w(y_1,\dots,y_t) &= \mathbb{I}(y_{t-n+1}\dots y_t = w) \\ G_u(y_1,\dots,y_t) &= \mathbb{I}(z_{k,t-n+1}\dots z_{k,t} = u) \end{aligned}$$

Once a first round of feature selection has been performed,[10] we compute $A[\mathcal{W}]$ as explained above. The last step of training reintroduces the observations and estimates the CRF paramaters. A variant of this approach adds extra *gappy* features to the n-gram features. Gappy features at order p test whether some label u occurs in the remote past anywhere between position $t-p+1$[11] and $t-n$. They take the following form:

$$\begin{aligned} G_{w,u}(y_1,\dots,y_t) = \mathbb{I}(&y_{t-n+1}\dots y_t = w \wedge \\ &u \in \{y_{t-p+1}\dots y_{t-n}\}), \end{aligned}$$

and likewise for features testing components.

4 Experiments

4.1 Training protocol

The following protocol is used throughout: (a) identify $\mathcal{W}$ (§3) - note that this may imply to tune a regularization parameter; (b) train a full model (including tests on the observations for each pattern in $\mathcal{W}$) using ℓ_1 regularization and a very small ℓ_2 term to stabilize convergence. The best regularization in (a) and (b) is selected on development data and targets either perplexity (for LMs) or label accuracy (for CRFs).

4.2 Datasets and Features

Experiments are run on two MRLs: for Czech, we use the CoNLL 2009 data set (Hajič et al., 2009) and for German, the Tiger Treebank with the split of Fraser et al. (2013)). Both datasets include rich morphological attributes (cf. Table 1).

All the patterns in $\mathcal{W}$ are combined with *lexical* features testing the current word x_t, its prefixes and suffixes of length 1 to 4, its capitalization and the presence of digit or punctuation symbols. Additional contextual features also test words in a local window around position t. These tests greatly increase the feature count and are not provided for all label patterns: for unigram patterns, we test the presence of all unigrams and bigrams of words in a window of 5 words; for bigrams patterns we only test for all unigrams in a window of 3 words. Contextual features are not used for larger patterns.

4.3 Results

We consider several baselines: Maxent and MEMM models, neither of which considers label dependencies in training, a linear chain CRF[12] and our own implementation of the group lasso of Vieira et al. (2016). For the latter, we contrast two setups: one where each pattern in $\mathcal{W}$ gives rise to one single feature, and one where it is conjoined with tests on the observation.[13] All scores in Table 2 are label accuracies on unseen test data.

As expected, Maxent and MEMM are outperformed by almost all variants of CRFs, and their scores are only reported for completeness. `Group lasso` results demonstrate the effectiveness of using contextual information with high order features: the gain is $\approx$ 0.7 points for both languages and all values of p. `Greedy` ℓ_1 achieves accuracy results similar to `group lasso`, suggesting that ℓ_1 penalty alone is effective to select high-order features. It also yields slighly smaller models and very comparable training time across the board: indeed, greedy parameter selection strategies imply multiple rounds of training which are overall quite costly, due to the size of the full label set. Testing individual subtags (§ 3.2) results in a slight improvement ($\approx$+0.3) in accuracy over `Greedy` ℓ_1. When using an additional automata for the full tag, we get a larger gain of $\approx$ 0.6 points for Czech, slightly less for German: including a model for complete tags also prevents to gener-

[10] As the LM building step only look at labels, we tune the regularization to optimize the perplexity of the LM on a development set.

[11] We use $p = 6$ in our experiments.

[12] Using the implementation of Lavergne et al. (2010).

[13] As suggested by the authors themselves in fn 4.

	cz				de			
	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
Maxent	90.01% 1924 191min	91.12% 1924 219min	91.17% 1924 286min	91.14% 1924 349min	85.62% 781 142min	85.84% 781 193min	85.96% 781 252min	86.02% 781 297min
MEMM	90.96% 1924 191min	92.09% 1924 219min	92.13% 1924 286min	92.12% 1924 349min	86.48% 781 142min	86.88% 781 193min	87.13% 781 252min	87.19% 781 297min
Linear Chain CRF	91.93% 3.7e6 657min	–	–	–	86.95% 6.1e5 447min	–	–	–
Group lasso	91.91% 9.6e5 421min	92.27% 4.3e7 1656min	92.41% 1.2e8 3067min	–	86.92% 1.8e5 305min	87.24% 9.2e6 1134min	87.48% 5.4e7 2101min	–
Group lasso + ctx	92.51% 9.2e5 520min	92.95% 4.1e7 1632min	93.03% 1.2e8 3285min	–	87.48% 1.7e5 349min	87.92% 7.8e6 1218min	87.96% 5.3e7 2398min	–
Greedy ℓ_1	92.47% 8.4e5 462min	92.94% 4.1e7 1759min	93.01% 1.1e8 3300min	–	87.43% 1.7e5 340min	87.87% 7.1e6 1239min	87.96% 5.0e7 2357min	–
Component-wise	92.76% 6.2e4 247min	93.24% 2.8e5 370min	93.36% 8.2e5 1179min	93.28% 3.7e6 2224min	87.47% 2.4e4 173min	88.16% 7.2e4 268min	88.26% 3.7e5 836min	88.29% 1.4e6 1483min
Component-wise + Full	92.97% 8.7e5 463min	93.41% 2.8e7 1569min	93.69% 8.3e7 3162min	93.65% 4.6e8 4321min	87.39% 1.4e5 311min	88.36% 5.2e6 1097min	88.59% 2.1e7 2181min	88.60% 1.3e8 3249min
Pruned LM	92.98% 3.2e5 233min	93.27% 8.2e6 487min	93.51% 1.1e7 1210min	93.53% 8.6e7 2519min	87.43% 1.3e5 163min	88.12% 8.9e5 372min	88.25% 9.1e6 896min	88.21% 5.3e7 1894min
MELM	93.02% 4.6e5 303min	93.33% 1.7e7 545min	93.81% 2.3e7 1478min	93.63% 1.4e8 2559min	87.41% 1.4e5 206min	88.61% 2.9e6 407min	88.76% 1.4e7 1063min	88.74% 9.8e7 1924min
MELM + Gaps	93.52% 4.5e5 289min	93.68% 1.5e7 658min	93.79% 1.9e7 1751min	–	88.38% 1.4e5 217min	88.70% 2.3e6 439min	88.78% 1.1e7 1297min	–

Table 2: Experimental results. Each cell reports accuracy, number of states in $A[\mathcal{W}]$ and total training time. Group lasso is our reimplementation of Vieira et al. (2016) (+Ctx = +context features) ; Greedy ℓ_1 is described in section 3.1, Component-wise is the decomposition approach of § 3.2, PrunedLM and MELM (+Gaps) were described in § 3.3 and § 3.4.

ate invalid combinations of subtags. These models represent different tradeoffs between accuracy and training time: the 4-gram `Component-wise` experiment only took 14 hrs to complete on German data and outperforms the corresponding `Greedy` ℓ_1 setup while containing approximately 100 times less features. `Component-wise+Full` is more comparable in size and training time to `Greedy` ℓ_1, but yields a larger improvement in performance. The last sets of experiments with LMs yields even better operating points, as the first stage of pattern selection is performed with a cheap model. They are our best trade-off to date, yielding the best performance for all values of p.

5 Conclusion

In this work, we have explored ways to take advantage of the flexibility offered by implementations of VoCRFs based on finite-state techniques. We have proposed strategies to include tests on subparts of complex tags, as well as to select useful label patterns with auxiliary unconditional LMs. Experiments with two MRLs with large tagsets yielded consistent improvements ($\approx$ +0.8 points) over strong baselines. They offer new perspectives to perform feature selection in high order CRFs. In our future work, we intend to also explore how to complement ℓ_1 penalties with terms penalizing more explicitely the processing time; we also wish to study how these ideas can be used in combination with neural models.

Acknowledgements

The authors wish to thank the reviewers for their useful comments and suggestions. This work has been partly funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 645452 (QT21).

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sapporo, Japan, pages 40–47. https://doi.org/10.3115/1075096.1075102.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.* 22(1):39–71.

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*. pages 24–41.

Ryan Cotterell and Jason Eisner. 2015. Penalized expectation propagation for graphical models over strings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 932–942. https://doi.org/10.3115/v1/N15-1094.

Nguyen Viet Cuong, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. 2014. Conditional Random Field with High-order Dependencies for Sequence Labeling and Segmentation. *Journal of Machine Learning Research* 15:981–1009. http://jmlr.org/papers/v15/cuong14a.html.

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. *CL* 39(1):57–85.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*. Seattle, WA, pages 94–101.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*. CoNLL '09, pages 1–18.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, Williamstown, MA, (ICML'01), pages 282–289.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden, pages 504–513.

Andre Martins, Noah Smith, Mario Figueiredo, and Pedro Aguiar. 2011. Structured sparsity in structured prediction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. pages 1500–1511.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, EMNLP'15, pages 2268–2274.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA, EMNLP'15, pages 322–332.

Chris Pal, Charles Sutton, and Andrew McCallum. 2006. Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*. volume 5, pages V–V. https://doi.org/10.1109/ICASSP.2006.1661342.

Dana Ron, Yoram Singer, and Naftali Tishby. 1996. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning* 25(2-3):117–149.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical learning modeling. *Computer, Speech and Language* 10:187 – 228.

Mark W. Schmidt and Kevin P. Murphy. 2010. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics,*. Chia Laguna Resort, Sardinia, Italy, AISTATS, pages 709–716.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1703–1714.

Hinrich Schütze and Yoram Singer. 1994. Part-of-speech tagging using a variable memory Markov model. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*. Las Cruces, New Mexico, pages 181–187.

Nobuyuki Shimizu and Andrew Haas. 2006. Exact decoding for jointly labeling and chunking sequences. In *Proceedings of COLING/ACL*. pages 763–770.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada, pages 475–482.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*. Lansdowne, VA, pages 270–274.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Langage Processing (ICSLP)*. Denver, CO, volume 2, pages 901–904.

Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. The MIT Press, Cambridge, MA.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B* 58(1):267–288.

Kristina Toutanova and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, pages 486–494. http://aclweb.org/anthology/P09-1055.

Tim Vieira, Ryan Cotterell, and Jason Eisner. 2016. Speed-accuracy tradeoffs in tagging with variable-order crfs and structured sparsity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 1973–1978.

Nan Ye, Wee S. Lee, Hai L. Chieu, and Dan Wu. 2009. Conditional random fields with high-order features for sequence labeling. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., pages 2196–2204. http://papers.nips.cc/paper/3815-conditional-random-fields-with-high-order-features-for-sequence-labeling.pdf.