Journal of Machine Learning Research 15 (2014) 981-1009 Submitted 10/12; Revised 9/13; Published 3/14

Conditional Random Field with High-order Dependencies for Sequence Labeling and Segmentation

Nguyen Viet Cuong NVCUONG@COMP.NUS.EDU.SG
Nan Ye YENAN@COMP.NUS.EDU.SG
Wee Sun Lee LEEWS@COMP.NUS.EDU.SG
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417

Hai Leong Chieu CHAILEON@DSO.ORG.SG
DSO National Laboratories
20 Science Park Drive
Singapore 118230

Editor: Kevin Murphy

Abstract

Dependencies among neighboring labels in a sequence are important sources of information for sequence labeling and segmentation. However, only first-order dependencies, which are dependencies between adjacent labels or segments, are commonly exploited in practice because of the high computational complexity of typical inference algorithms when longer distance dependencies are taken into account. In this paper, we give efficient inference algorithms to handle high-order dependencies between labels or segments in conditional random fields, under the assumption that the number of distinct label patterns used in the features is small. This leads to efficient learning algorithms for these conditional random fields. We show experimentally that exploiting high-order dependencies can lead to substantial performance improvements for some problems, and we discuss conditions under which high-order features can be effective.

Keywords: conditional random field, semi-Markov conditional random field, high-order feature, sequence labeling, segmentation, label sparsity

1. Introduction

Many problems can be cast as the problem of labeling or segmenting a sequence of observations. Examples include natural language processing tasks, such as part-of-speech tagging (Lafferty et al., 2001), phrase chunking (Sha and Pereira, 2003), named entity recognition (McCallum and Li, 2003), and tasks in bioinformatics such as gene prediction (Culotta et al., 2005) and RNA secondary structure prediction (Durbin, 1998).

Conditional random field (CRF) (Lafferty et al., 2001) is a discriminative, undirected Markov model which represents a conditional probability distribution of a structured output variable $\mathbf{y}$ given an observation $\mathbf{x}$. Conditional random fields have been successfully applied in sequence labeling and segmentation. Compared to generative models such as hidden Markov models (Rabiner, 1989), CRFs model only the conditional distribution of $\mathbf{y}$

Type of CRF	Feature example
First-order (Lafferty et al., 2001)	*author year*
Semi-CRF (Sarawagi and Cohen, 2004)	*author+ year+*
High-order (Ye et al., 2009, this paper)	*author year title title*
High-order semi-CRF (this paper)	*author+ year+ title+*

Table 1: Examples of the information that can be captured by different types of CRFs for the bibliography extraction task. The $x+$ symbol represents a segment of "1 or more" labels of class x.

given $\mathbf{x}$, and do not model the observations $\mathbf{x}$. Hence, CRFs can be used to encode complex dependencies of $\mathbf{y}$ on $\mathbf{x}$ without significantly increasing the inference and learning costs. However, inference for CRFs is NP-hard in general (Istrail, 2000), and most CRFs have been restricted to consider very local dependencies. Examples include the linear-chain CRF which considers dependencies between at most two adjacent labels (Lafferty et al., 2001) and the first-order semi-Markov CRF (semi-CRF) which considers dependencies between at most two adjacent segments (Sarawagi and Cohen, 2004), where a segment is a contiguous sequence of identical labels. In linear-chain CRF and semi-CRF, a k^{th}-order feature is a feature that encodes the dependency between $\mathbf{x}$ and $(k+1)$ consecutive labels or segments. Existing inference algorithms for CRFs such as the Viterbi and the forward-backward algorithms can only handle up to first-order features, and inference algorithms for semi-CRFs (Sarawagi and Cohen, 2004) can only handle up to first-order features between segments. These algorithms can be easily generalized to handle high-order features, but will require time exponential in k. In addition, a general inference algorithm such as the clique tree algorithm (Huang and Darwiche, 1996) also requires time exponential in k to handle k^{th}-order features ($k > 1$).

In this paper, we exploit a form of sparsity that is often observed in real data to design efficient algorithms for inference and learning with high-order label or segment dependencies. Our algorithms are presented for high-order semi-CRFs in its most general form. Algorithms for high-order CRFs are obtained by restricting the segment lengths to 1, and algorithms for linear-chain CRFs and first-order semi-CRFs are obtained by restricting the maximum order to 1.

We use a bibliography extraction task in Table 1 to show examples of features that can be used with different classes of CRFs. In this task, different fields are often arranged in a fixed order, hence using high-order features can be advantageous. The sparsity property that we exploit is the following *label pattern sparsity*: the number of observed sequences of k consecutive segment labels (e.g., "*author+ year+ title+*" is one such sequence where $k = 3$) is much smaller than n^k, where n is the number of distinct labels. This assumption often holds in real problems. Under this assumption, we give algorithms for computing marginals, partition functions, and Viterbi parses for high-order semi-CRFs. The partition function and the marginals can be used to efficiently compute the log-likelihood and its gradient. In turn, the log-likelihood and its gradient can be used with quasi-Newton methods to efficiently find the maximum likelihood parameters (Sha and Pereira, 2003). The algo-

rithm for Viterbi parsing can also be used with cutting plane methods to train max-margin solutions for sequence labeling problems in polynomial time (Tsochantaridis et al., 2004). Our inference and learning algorithms run in time polynomial in the maximum segment length as well as the number and length of the label patterns that the features depend on.

We demonstrate that modeling high-order dependencies can lead to significant performance improvements in various problems. In our first set of experiments, we focus on high-order CRFs and demonstrate that using high-order features can improve performance in sequence labeling problems. We show that in handwriting recognition, using even simple high-order indicator features improves performance over using linear-chain CRFs, and significant performance improvement is observed when the maximum order of the indicator features is increased. We also use a synthetic data set to discuss the conditions under which high-order features can be helpful. In our second set of experiments, we demonstrate that using high-order semi-Markov features can be helpful in some applications. More specifically, we show that high-order semi-CRFs outperform high-order CRFs and first-order semi-CRFs on three segmentation tasks: relation argument detection, punctuation prediction, and bibliography extraction.[1]

2. Algorithms for High-order Dependencies

Our algorithms are presented for high-order semi-CRFs in its most general form. These algorithms generalize the algorithms for linear-chain CRFs and first-order semi-CRFs, which are special cases of our algorithms when the maximum order is set to 1. They also generalize the algorithms for high-order CRFs (Ye et al., 2009), which are special cases of our algorithms when the segment lengths are set to 1. Thus, only the general algorithms described in this section need to be implemented to handle all these different cases.[2]

2.1 High-order Semi-CRFs

Let $\mathcal{Y} = \{1, 2, \ldots, n\}$ denote the set of distinct labels, $\mathbf{x} = (x_1, \ldots, x_{|\mathbf{x}|})$ denote an input sequence of length $|\mathbf{x}|$, and $\mathbf{x}_{a:b}$ denote the sub-sequence $(x_a, \ldots, x_b)$. A *segment* of $\mathbf{x}$ is defined as a triplet (u, v, y), where y is the common label of the segment $\mathbf{x}_{u:v}$. A *segmentation* for $\mathbf{x}_{a:b}$ is a segment sequence $\mathbf{s} = (s_1, \ldots, s_p)$, with $s_j = (u_j, v_j, y_j)$ such that $u_{j+1} = v_j + 1$ for all j, $u_1 = a$ and $v_p = b$. A segmentation for $\mathbf{x}_{a:b}$ is a partial segmentation for $\mathbf{x}$.

A semi-CRF defines a conditional distribution over all possible segmentations $\mathbf{s}$ of an input sequence $\mathbf{x}$ such that

$$P(\mathbf{s}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \exp\Big(\sum_{i=1}^{m} \sum_{t=1}^{|\mathbf{s}|} \lambda_i f_i(\mathbf{x}, \mathbf{s}, t)\Big)$$

1. This paper is an extended version of a previous paper (Ye et al., 2009) published in NIPS 2009. Some of the additional material presented here has also been presented as an abstract (Nguyen et al., 2011) at the ICML Workshop on Structured Sparsity: Learning and Inference, 2011. The source code for our algorithms is available at `https://github.com/nvcuong/HOSemiCRF`.
2. In an earlier paper (Ye et al., 2009), we give algorithms for high-order CRFs which are similar to those presented here. The main difference lies in the backward algorithm. The version presented here is a *conditional* version which uses properties of labels before the suffix labels being considered, making extension to the high-order semi-Markov features simpler.

where $Z_{\mathbf{x}} = \sum_{\mathbf{s}} \exp(\sum_i \sum_t \lambda_i f_i(\mathbf{x}, \mathbf{s}, t))$ is the partition function with the summation over all segmentations of $\mathbf{x}$, and $\{f_i(\mathbf{x}, \mathbf{s}, t)\}_{1 \leq i \leq m}$ is the set of semi-Markov features, each of which has a corresponding weight λ_i.

We shall work with features of the following form

$$f_i(\mathbf{x}, \mathbf{s}, t) = \begin{cases} g_i(\mathbf{x}, u_t, v_t) & \text{if } y_{t-|\mathbf{z}^i|+1} \dots y_t = \mathbf{z}^i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $\mathbf{z}^i \in \mathcal{Y}^{|\mathbf{z}_i|}$ is a segment label pattern associated with f_i, and $\mathbf{s}$ is a segmentation or a partial segmentation for $\mathbf{x}$. The function $f_i(\mathbf{x}, \mathbf{s}, t)$ depends on the t-th segment as well as the label pattern $\mathbf{z}^i$ and is said to be of order $|\mathbf{z}^i| - 1$. The order of the resulting semi-CRF is the maximal order of the features.

We will give exact inference algorithms for high-order semi-CRFs in the following sections. As in exact inference algorithms for linear-chain CRFs and semi-CRFs, our algorithms perform forward and backward passes to obtain the necessary information for inference.

2.2 Notations

Without loss of generality, let $\mathcal{Z} = \{\mathbf{z}^1, \dots, \mathbf{z}^M\}$ be the *segment label pattern set*, that is, the set of distinct segment label patterns of the m features ($M \leq m$). For our forward algorithm, the *forward-state set* $\mathcal{P} = \{\mathbf{p}^1, \dots, \mathbf{p}^{|\mathcal{P}|}\}$ consists of distinct elements in the set of all the labels and proper prefixes (including the empty sequence ϵ) of the segment label patterns. Thus, $\mathcal{P} = \mathcal{Y} \cup \{\mathbf{z}^j_{1:k}\}_{0 \leq k < |\mathbf{z}^j|, 1 \leq j \leq M}$. For the backward algorithm, the *backward-state set* $\mathcal{S} = \{\mathbf{s}^1, \dots, \mathbf{s}^{|\mathcal{S}|}\}$ consists of distinct elements in $\mathcal{P}\mathcal{Y}$, that is, the set consisting of elements in $\mathcal{P}$ concatenated with a label in $\mathcal{Y}$.

Transitions between states in our algorithm are defined using the suffix relationships between them. We use $\mathbf{z}_1 \leq^s \mathbf{z}_2$ to denote that $\mathbf{z}_1$ is a suffix of $\mathbf{z}_2$. The longest suffix relation on a set $\mathcal{A}$ is denoted by $\mathbf{z}_1 \leq^s_{\mathcal{A}} \mathbf{z}_2$. This relation holds true if and only if $\mathbf{z}_1$, among all the elements of $\mathcal{A}$, is the longest suffix of $\mathbf{z}_2$. More formally, $\mathbf{z}_1 \leq^s_{\mathcal{A}} \mathbf{z}_2$ if and only if $\mathbf{z}_1 \in \mathcal{A}$ and $\mathbf{z}_1 \leq^s \mathbf{z}_2$ and $\forall \mathbf{z} \in \mathcal{A}, \mathbf{z} \leq^s \mathbf{z}_2 \Rightarrow \mathbf{z} \leq^s \mathbf{z}_1$.

2.3 Training

Given a training set $\mathcal{T}$, we estimate the model parameters $\vec{\lambda} = (\lambda_1, \dots, \lambda_m)$ by maximizing the regularized log-likelihood function

$$\mathcal{L}_{\mathcal{T}}(\vec{\lambda}) \quad = \quad \sum_{(\mathbf{x},\mathbf{s}) \in \mathcal{T}} \log P(\mathbf{s}|\mathbf{x}) - \sum_{i=1}^{m} \frac{\lambda_i^2}{2\sigma^2_{\text{reg}}}$$

where σ_{reg} is a regularization parameter. This function is convex, and thus can be maximized using any convex optimization algorithm. In our implementation, we use the L-BFGS method (Liu and Nocedal, 1989). The method requires computation of the value of $\mathcal{L}_{\mathcal{T}}(\vec{\lambda})$ and its partial derivatives

$$\frac{\partial \mathcal{L}_{\mathcal{T}}}{\partial \lambda_i} = \tilde{E}(f_i) - E(f_i) - \frac{\lambda_i}{\sigma^2_{\text{reg}}}$$

where $\tilde{E}(f_i) = \sum_{(\mathbf{x},\mathbf{s}) \in \mathcal{T}} \sum_t f_i(\mathbf{x}, \mathbf{s}, t)$ is the empirical feature sum of the feature f_i, and $E(f_i) = \sum_{(\mathbf{x},\mathbf{s}) \in \mathcal{T}} \sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{x}) \sum_t f_i(\mathbf{x}, \mathbf{s}', t)$ is the expected feature sum of f_i. To compute

$\mathcal{L}_{\mathcal{T}}(\vec{\lambda})$ and its partial derivatives, we need to efficiently compute the partition function $Z_{\mathbf{x}}$ and the expected feature sum of f_i's.

2.3.1 Partition Function

For any $\mathbf{p}^i \in \mathcal{P}$, let $\mathbf{p}_{j,\mathbf{p}^i}$ be the set of all segmentations for $\mathbf{x}_{1:j}$ whose segment label sequences contain $\mathbf{p}^i$ as the longest suffix among all elements in $\mathcal{P}$. We define the forward variables $\alpha_{\mathbf{x}}(j, \mathbf{p}^i)$ as follows

$$\alpha_{\mathbf{x}}(j, \mathbf{p}^i) = \sum_{\mathbf{s} \in \mathbf{p}_{j,\mathbf{p}^i}} \exp(\sum_k \sum_t \lambda_k f_k(\mathbf{x}, \mathbf{s}, t)).$$

The above definition of the forward variable $\alpha_{\mathbf{x}}$ is the same as the usual definition of forward variable for first-order semi-CRFs when only zeroth-order and first-order semi-Markov features are used. The forward variables can be computed by dynamic programming:

$$\alpha_{\mathbf{x}}(j, \mathbf{p}^i) = \sum_{d=0}^{L-1} \sum_{(\mathbf{p}^k, y): \mathbf{p}^i \leq_{\mathcal{P}}^{s} \mathbf{p}^k y} \Psi_{\mathbf{x}}(j-d, j, \mathbf{p}^k y) \alpha_{\mathbf{x}}(j-d-1, \mathbf{p}^k)$$

where L is the longest possible length of a segment, $\sum_{i:\mathrm{Pred}(i)}$ denotes summation over all i's satisfying the predicate $\mathrm{Pred}(i)$, and $\Psi_{\mathbf{x}}(u, v, \mathbf{p})$ counts the contribution of features activated when there is a segment label sequence $\mathbf{p}$ with its last segment having boundary (u, v). The factor $\Psi_{\mathbf{x}}(u, v, \mathbf{p})$ is defined as

$$\Psi_{\mathbf{x}}(u, v, \mathbf{p}) = \exp(\sum_{i: \mathbf{z}^i \leq^s \mathbf{p}} \lambda_i g_i(\mathbf{x}, u, v)).$$

The correctness of the above recurrence is shown in Appendix A. The partition function can be computed from the forward variables by

$$Z_{\mathbf{x}} = \sum_i \alpha_{\mathbf{x}}(|\mathbf{x}|, \mathbf{p}^i).$$

2.3.2 Expected Feature Sum

Let $\mathbf{s}_j$ be the set of all partial segmentations for $\mathbf{x}_{j:|\mathbf{x}|}$. For $\mathbf{s} \in \mathbf{s}_j$ and $\mathbf{s}^k \in \mathcal{S}$, we define for each feature f_i a conditional feature function $f_i(\mathbf{x}, \mathbf{s}, t | \mathbf{s}^k)$, which takes the value of $f_i(\mathbf{x}, \mathbf{s}, t)$ when $\mathbf{s}^k$ is the longest suffix (in $\mathcal{S}$) of the segment label sequence for $\mathbf{x}_{1:j-1}$. Otherwise, its value is 0. For example, if $\mathbf{s} = (s_1, \ldots, s_p) \in \mathbf{s}_j$ and $s_1 = (u_1, v_1, y_1)$, then

$$f_i(\mathbf{x}, \mathbf{s}, 1 | \mathbf{s}^k) = \begin{cases} g_i(\mathbf{x}, u_1, v_1) & \text{if } \mathbf{z}^i \leq^s \mathbf{s}^k y_1 \\ 0 & \text{otherwise} \end{cases}.$$

For each $\mathbf{s}^i \in \mathcal{S}$, we define the backward variables $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$ as follows

$$\beta_{\mathbf{x}}(j, \mathbf{s}^i) = \sum_{\mathbf{s} \in \mathbf{s}_j} \exp(\sum_k \sum_t \lambda_k f_k(\mathbf{x}, \mathbf{s}, t | \mathbf{s}^i)).$$

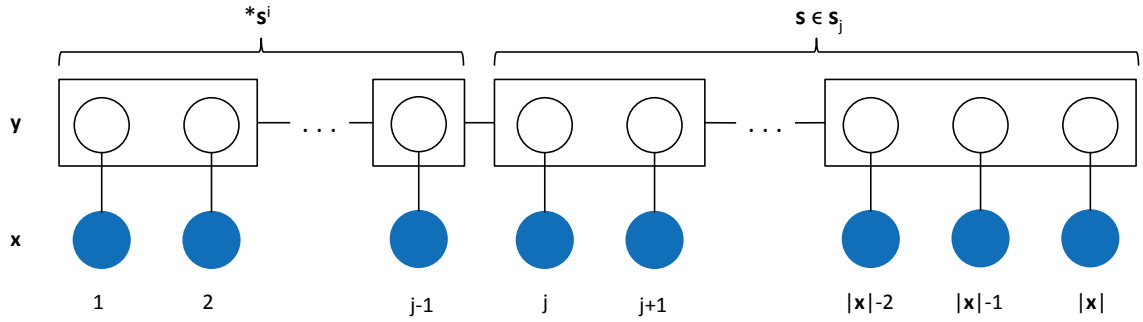


Figure 1: An illustration of the backward variable $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$. Each rectangular box corresponds to a segment. The regular expression $*\mathbf{s}^i$ means that $\mathbf{s}^i$ is the suffix of the segment label sequence for $\mathbf{x}_{1:j-1}$. In fact, $\mathbf{s}^i$ is the longest suffix of the segment label sequence for $\mathbf{x}_{1:j-1}$. The summation in the definition of $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$ is over all the partial segmentations $\mathbf{s}$ of $\mathbf{x}_{j:|\mathbf{x}|}$.

Figure 1 gives an illustration of the backward variable $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$. Note that our definition of $\beta_{\mathbf{x}}$ uses the conditional feature function and does not generalize the usual definitions of the backward variables in first-order semi-CRFs (Sarawagi and Cohen, 2004) or high-order CRFs (Ye et al., 2009).

Similar to the case of forward variables, we can compute $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$ by dynamic programming:

$$\beta_{\mathbf{x}}(j, \mathbf{s}^i) = \sum_{d=0}^{L-1} \sum_{(\mathbf{s}^k, y): \mathbf{s}^k \leq_S^s \mathbf{s}^i y} \Psi_{\mathbf{x}}(j, j+d, \mathbf{s}^i y) \beta_{\mathbf{x}}(j+d+1, \mathbf{s}^k).$$

In Appendix A, we show the correctness proof for the recurrence. We can now compute the marginals $P(u, v, \mathbf{z}|\mathbf{x})$ for each $\mathbf{z} \in \mathcal{Z}$ and $u \leq v$, where $P(u, v, \mathbf{z}|\mathbf{x})$ denotes the probability that a segmentation of $\mathbf{x}$ contains label pattern $\mathbf{z}$ and has (u, v) as $\mathbf{z}$'s last segment boundaries. These marginals can be computed by

$$P(u, v, \mathbf{z}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \sum_{(\mathbf{p}^i, y): \mathbf{z} \leq^s \mathbf{p}^i y} \alpha_{\mathbf{x}}(u-1, \mathbf{p}^i) \Psi_{\mathbf{x}}(u, v, \mathbf{p}^i y) \beta_{\mathbf{x}}(v+1, \mathbf{p}^i y).$$

We compute the expected feature sum for f_i by

$$E(f_i) = \sum_{(\mathbf{x}, \mathbf{s}) \in \mathcal{T}} \sum_{u \leq v} P(u, v, \mathbf{z}^i|\mathbf{x}) g_i(\mathbf{x}, u, v).$$

In Appendix B, we give an example to illustrate our algorithms for the second-order CRF model.

Using the conditional feature function to define the backward variables $\beta_{\mathbf{x}}$ can help to simplify the computation of the marginals for high-order semi-CRF models. If we directly generalized the usual definition of the backward variables (Ye et al., 2009) to high-order semi-CRFs (which can be done easily), computing the marginals using these backward variables would be complicated. The main reason is that the semi-Markov features in Equation

(1) only know the correct position (u_t, v_t) of the last segment. In other words, although they know the label sequence of the previous segments, the features do not know the actual boundaries of these segments. So, to compute the marginal $P(u, v, \mathbf{z}|\mathbf{x})$ using the usual extension of the backward variables, we need to sum over all possible segmentations near (u, v) that contain (u, v) as a segment. This may result in an algorithm that is exponential in the order of the semi-CRFs. Note that this problem does not occur for high-order CRFs (Ye et al., 2009) since in these models, the segment length is 1 and thus we can always determine the boundaries of the segments.

2.4 Decoding

We compute the most likely segmentation for high-order semi-CRF by a Viterbi-like decoding algorithm. It is the same as the forward algorithm with the sum operator replaced by the max operator. Define

$$\delta_{\mathbf{x}}(j, \mathbf{p}^i) = \max_{\mathbf{s} \in \mathbf{P}_{j,\mathbf{p}^i}} \exp(\sum_k \sum_t \lambda_k f_k(\mathbf{x}, \mathbf{s}, t)).$$

These variables can be computed by

$$\delta_{\mathbf{x}}(j, \mathbf{p}^i) = \max_{(d,\mathbf{p}^k,y):\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{p}^k y} \Psi_{\mathbf{x}}(j-d, j, \mathbf{p}^k y)\delta_{\mathbf{x}}(j-d-1, \mathbf{p}^k).$$

Note that the value of d is inclusively between 0 and $L-1$ in the above equation. The most likely segmentation can be obtained using backtracking from $\max_i \delta_{\mathbf{x}}(|\mathbf{x}|, \mathbf{p}^i)$.

2.5 Time Complexity

We now give rough time bounds for the above algorithm. It is important to note that the bounds given in this part are pessimistic, and the computation can be done more quickly in practice. For simplicity, we assume that the features $g_i(\cdot, \cdot, \cdot)$ can be computed in $O(1)$ time for all $i \in \{1, 2, \ldots, m\}$ and the algorithm would pre-compute all the values of $\Psi_{\mathbf{x}}$ before doing the forward and backward passes. This assumption often holds for features used in practice, although one can define g_i's which are arbitrarily difficult to compute.

Since the total number of different patterns of the last argument of $\Psi_{\mathbf{x}}$ is $O(|\mathcal{S}||\mathcal{Y}|) = O(|\mathcal{P}||\mathcal{Y}|^2)$, the time complexity to pre-compute all the values of $\Psi_{\mathbf{x}}$ in the worst case is $O(mT^2|\mathcal{P}||\mathcal{Y}|^2) = O(mn^2T^2|\mathcal{P}|)$, where T is the maximum length of an input sequence. After pre-computing the values of $\Psi_{\mathbf{x}}$, we can compute all the values of $\alpha_{\mathbf{x}}$ in $O(T^2|\mathcal{Y}||\mathcal{P}|)$ time. Similarly, the time complexity to compute all the values of $\beta_{\mathbf{x}}$ is $O(T^2|\mathcal{Y}||\mathcal{S}|)$. Then, with these values, we can compute all the marginal probabilities in $O(T^2|\mathcal{Z}||\mathcal{P}|)$. Finally, the time complexity for decoding is $O(T^2|\mathcal{Y}||\mathcal{P}|)$.

3. Experiments

In this section, we describe experiments comparing CRFs, semi-CRFs, high-order CRFs, and high-order semi-CRFs. The experiments in Section 3.1 show the advantages of the high-order CRFs, while those in Section 3.2 show the advantages of the high-order semi-CRFs.

3.1 Experiments with High-order CRFs

The practical feasibility of making use of high-order features based on our algorithm lies in the observation that the label pattern sparsity assumption often holds. Our algorithm can be applied to take those high-order features into consideration: high-order features now form a component that one can play with in feature engineering.

Now, the question is whether high-order features are *practically significant*. We first use a synthetic data set to explore conditions under which high-order features can be expected to help. We then use a handwritten character recognition problem to demonstrate that even incorporating simple high-order features can lead to impressive performance improvement on a naturally occurring data set.[3]

3.1.1 Synthetic Data Generated Using k^{th}-order Markov Model

We randomly generate an order k Markov model with n states $s_1, \ldots, s_n$ as follows. To increase pattern sparsity, we allow at most r randomly chosen possible next states given the previous k states. This limits the number of possible label sequences in each length $(k+1)$ segment from n^{k+1} to $n^k r$. The conditional probabilities of these r next states are generated by randomly selecting a vector from the uniform distribution over $[0,1]^r$ and normalizing them. Each state s_i generates an observation $(a_1, \ldots, a_m)$ such that a_j follows a Gaussian distribution with mean μ_{ij} and standard deviation σ. Each $\mu_{i,j}$ is independently randomly generated from the uniform distribution over $[0,1]$. In the experiments, we use values of $n = 5$, $r = 2$ and $m = 3$.

The standard deviation σ controls how much information the observations reveal about the states. If σ is very small as compared to most μ_{ij}'s, then using the observations alone as features is likely to be good enough to obtain a good classifier of the states; the label correlations become less important for classification. However, if σ is large, then it is difficult to distinguish the states based on the observations alone and the label correlations, particularly those captured by higher order features are likely to be helpful.

We use the current, previous, and next observations, rather than just the current observation as features, exploiting the conditional probability modeling strength of CRFs. For higher order features, we simply use all indicator features that appeared in the training data up to a maximum order. We considered the case $k = 2$ and $k = 3$, and varied σ and the maximum order. We run the experiment with training sets that contain 300, 400, and 500 sequences, and evaluate the models on a test set that contains 500 sequences. All the sequences are of length 20; each sequence was initialized with a random sequence of length k and generated using the randomly generated order k Markov model. Training was done by maximizing the regularized log-likelihood with regularization parameter $\sigma_{\text{reg}} = 1$ in all experiments in this paper. The experimental results are shown in Figures 2.

Figure 2 shows that the high-order indicator features are useful in all cases. In particular, we can see that it is beneficial to increase the order of the high-order features when the underlying model has longer distance correlations. As expected, increasing the order of the features beyond the order of the underlying model is not helpful. The results also suggest

3. The results given in the earlier version of this work (Ye et al., 2009) are significantly lower than the results presented here due to a bug in the decoding algorithm. We have fixed the bug and reported the corrected results in this paper.

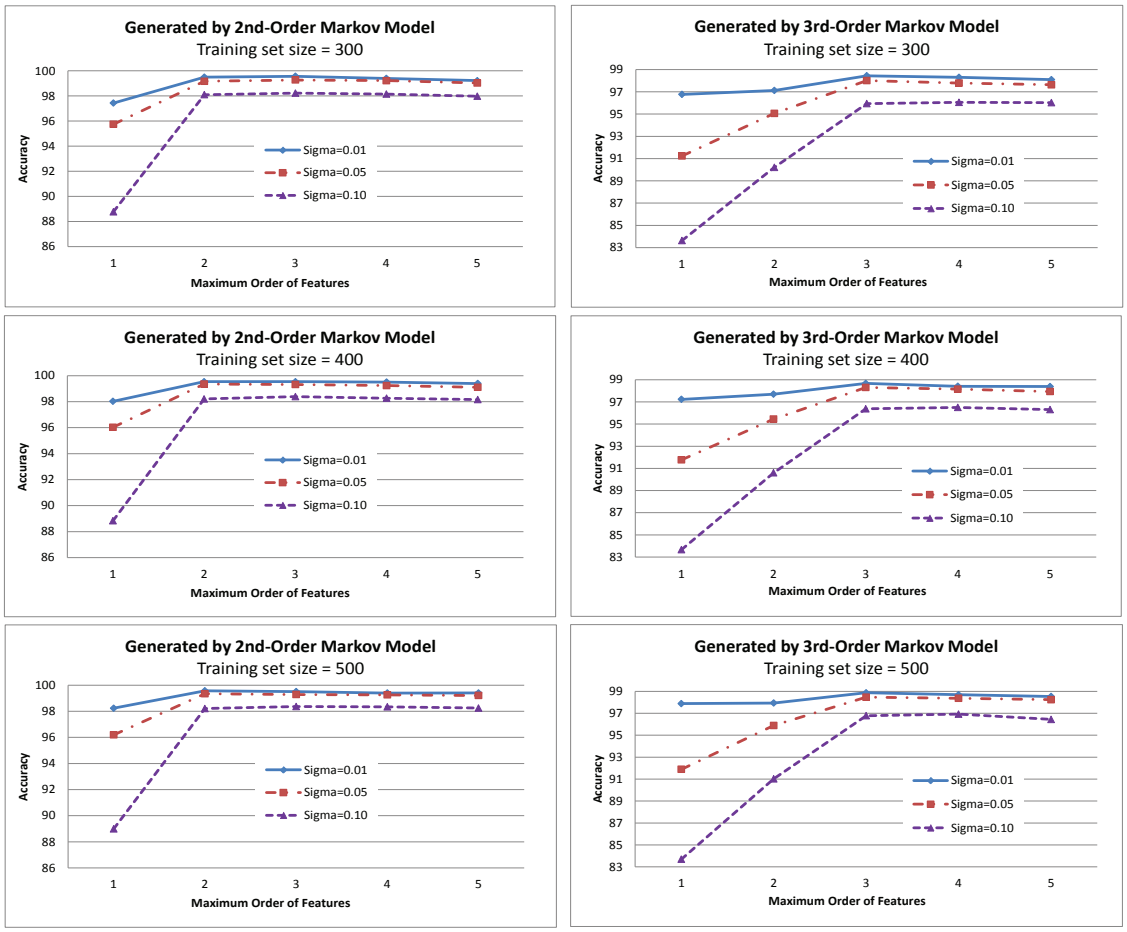


Figure 2: Accuracy of high-order CRFs as a function of maximum order on synthetic data sets.

that in general, if the observations are closely coupled with the states (in the sense that different states correspond to very different observations), then feature engineering on the observations is generally enough to perform well, and it is less important to use high-order features to capture label correlations. On the other hand, when such coupling is not clear, it becomes important to capture the label correlations, and high-order features can be useful.

We also study the effects of spurious, rare high-order patterns, and show that such patterns in the training or test set do not significantly impair performance of high-order CRFs in our experiments. For this purpose, we tabulate the proportion of fourth-order patterns (i.e., length 5 patterns) exclusive to the training or test sets in Table 2. The statistics show that around 10% of the patterns are exclusive to the training or test data. On the other hand, the results in Figure 2 show that when these patterns are used in the fourth-order model, the performance only drops slightly. Even if we increase the number of spurious, rare high-order patterns (by reducing the training data size), there is no significant drop in accuracies for high-order CRFs.

Size	300		400		500	
Order	Train	Test	Train	Test	Train	Test
2	16/173	13/170	17/175	12/170	17/177	10/170
3	34/393	58/417	37/406	48/417	42/424	35/417

Table 2: Proportions of length 5 patterns exclusive to training and test data where the data sets are generated by 2^{nd}-order and 3^{rd}-order Markov models. For each proportion, the denominator shows the number of patterns in the data set, and the numerator shows the number of patterns exclusive to it. Nearly all of these patterns occur for less than 5 times (mostly once or twice). Note that the labels are first generated independently of σ in our data sets, thus the statistics are the same for all σ values.

In practical problems, regularization may work well as a means for avoiding overfitting spurious high-order features. But this depends on how heavily the training process is regularized, and some tuning may be needed. For example, for a regularizer like Gaussian regularizer $\sum_i \frac{\lambda_i^2}{2\sigma_{reg}^2}$, the parameter σ_{reg} is often determined using a validation data set or cross-validation on the training data.

3.1.2 Handwriting Recognition

We used the handwriting recognition data set (Taskar et al., 2004), consisting of around 6100 handwritten words with an average length of around 8 characters. The data was originally collected by Kassel (1995) from around 150 human subjects. The words were segmented into characters, and each character was converted into an image of 16 by 8 binary pixels. In this labeling problem, each x_i is the image of a character, and each y_i is a lower-case letter. The experimental setup is the same as that used by Taskar et al. (2004): the data set was divided into 10 folds with each fold having approximately 600 training and 5500 test examples and the zero-th order features for a character are the pixel values.

For high-order features, we again used all indicator features that appeared in the training data up to a maximum order. The average accuracies over the 10 folds are shown in Figure 3, where strong improvements are observed as the maximum order increases. Figure 3 also shows the number of label patterns, the total training time, and the running time per iteration of the L-BFGS algorithm (which requires computation of the gradient and value of the function at each iteration). Both the number of patterns and the running time appear to grow no more than linearly with the maximum order of the features for this data set.

3.2 Experiments with High-order Semi-CRFs

We now show that high-order semi-CRFs are also practically useful by evaluating their performance on three different sequence labeling tasks: relation argument detection, punctuation prediction in movie transcripts, and bibliography extraction. We compare high-order semi-CRFs with CRFs of different orders on the same tasks. In our tables, C^k and SC^k

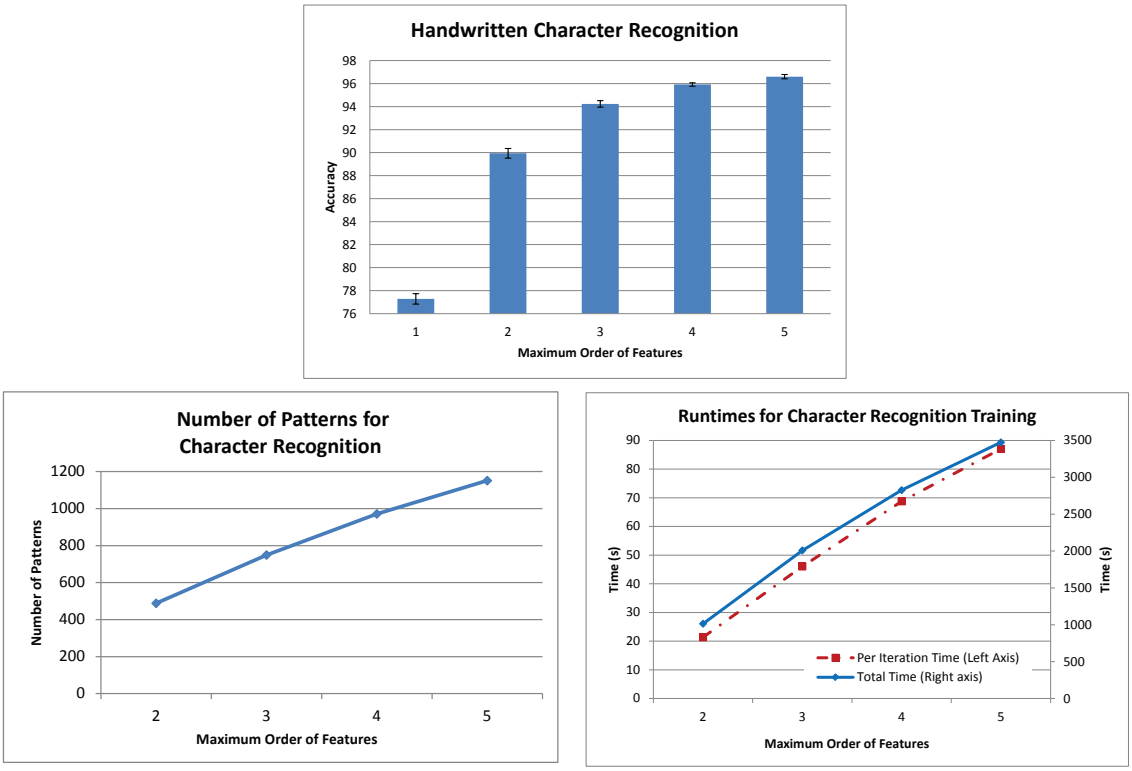


Figure 3: Accuracy (top), number of label patterns (bottom left), and running time (bottom right) as a function of maximum order for the handwriting recognition data set.

refer to k^{th}-order CRF and semi-CRF respectively. We also give the number of segment label patterns and the running time of high-order semi-CRFs on the tasks.

To test if the results obtained by high-order semi-CRFs are significantly better than lower order ones in terms of F1-measure, we perform the randomization tests described by Noreen (1989) and Yeh (2000). In such tests, we shuffle the responses by randomly reassigning the outputs of two systems we are comparing, and see how likely such a shuffle produces a difference in the metric of interest (in our case, the F1-measure). An exact randomization test will iterate through all possible shuffles, but due to the large data sizes, we use an approximate randomization test where for each comparison, we perform 10000 random shuffles, and we repeat this for 999 times. It can be shown (Noreen, 1989; Yeh, 2000) that the significance level p is at most $p' = (n_c+1)/(n_t+1)$, where n_c is the number of trials in which the difference between the F1-measures is greater than the original difference, and n_t is the total number of iterations (in our case, 999). In Table 4, 7, and 9, we summarize the p' obtained in the significance tests. We will comment on these results for each of the three data sets in the following sections.

3.2.1 Relation Argument Detection

In this experiment, we consider the problem of relation argument detection, which identifies and labels arguments of relations in English sentences. More specifically, we construct the label sequence for each sentence as follows: If a word in a sentence is the first argument of a relation, we label it as *Arg1*. If it is the second argument, we label it as *Arg2*. If the word is the first argument of a relation and it is also the second argument of another relation of the same type, we label it as *Arg1Arg2*. Otherwise, we label it as *O*, which means the word is not part of any relation. For example, in the labeled sentence "Peter/Arg1 is/O working/O for/O IBM/Arg2 ./O", *Peter* and *IBM* are arguments of a relation.

It is important to note that if a sentence contains many *Arg1*'s and *Arg2*'s, we do not know which pairs of *Arg1* and *Arg2* would be the actual arguments of a relation. Furthermore, the matching of *Arg1*'s and *Arg2*'s is not one-to-one either, since a word may participate in many different relations of the same type. Thus, to actually extract the relations in a sentence, we would need a separate classifier to determine which pairs of *Arg1* and *Arg2* are the true mentions of a relation. In this experiment, however, we only focus and report on the sentence labeling task.

The relation argument detection problem can be thought of as part of the relation extraction task, which requires extracting some prespecified relationships between named entity mentions. For example, if a person works for an organization, then the person and the organization form an organization-affiliation relation. Previous works on the relation extraction problem usually involve building a classifier to decide whether two named entity mentions are the actual arguments of the relation (GuoDong et al., 2005; Zhang et al., 2006). It may also be beneficial for the classifiers if they can make use of the information obtained from relation argument detection.

We compared the models on the English portion of the Automatic Content Extraction (ACE) 2005 corpus (Walker et al., 2006). The corpus contains articles from six source domains and we group the labeled relations into six types. For the experiment, we trained a separate tagger for each type of relations. The training set contains 70% of the sentences from each source domain. The remaining 30% of the sentences are used for testing. Most sentences do not contain a relation and they make the trained tagger less likely to predict an argument. Hence, we randomly sampled from these negative examples so that the numbers of positive and negative examples are the same. We also assumed the manually annotated named entity mentions are known.

For linear-chain CRF, the zeroth-order features are: surrounding words before and after the current word and their capitalization patterns; letter n-grams in words; surrounding named entity mentions, part-of-speeches before and after the current word and their combinations. The first-order features are: transitions without any observation, transitions with the current or previous words or combinations of their capitalization patterns. The high-order CRFs and semi-CRFs include additional high-order Markov and high-order semi-Markov transition features.

From the results in Table 3, SC^2 gives an improvement of 5.52% on F1 score when compared to SC^1 on average. SC^3 further improves the performance of SC^2 by 0.75% F1 score. High-order CRFs show significant improvement on all except for *PHYS*, which has arguments located further apart compared to other relations. In Table 4, we see that

Type	C^1	C^2	C^3	SC^1	SC^2	SC^3
Part-Whole	38.68	41.41	**46.52**	38.57	42.56	44.30
Phys	33.24	33.60	35.20	33.35	42.04	**42.46**
Org-Aff	60.56	63.28	**64.93**	60.77	63.72	64.86
Gen-Aff	31.00	35.84	**40.16**	31.19	35.85	38.09
Per-Soc	53.67	**58.62**	58.31	53.46	57.66	57.07
Art	40.30	43.80	46.35	40.61	**49.21**	48.78
Average	42.91	46.09	48.58	42.99	48.51	**49.26**

Table 3: F1 scores of different CRF taggers for relation argument detection on six types of relations.

	C^2	C^3	SC^1	SC^2	SC^3
C^1	**0.001**<	**0.001**<	0.226<	**0.001**<	**0.001**<
C^2	–	**0.001**<	**0.001**>	**0.001**<	**0.001**<
C^3	–	–	**0.001**>	0.441>	0.074<
SC^1	–	–	–	**0.001**<	**0.001**<
SC^2	–	–	–	–	0.017<

Table 4: The values of p' obtained in the statistical significance tests comparing CRFs and semi-CRFs of different orders in the relation argument detection task, where the p-value of the significance test is smaller than p'. Figures in bold are where the difference is statistically significant at the 1% confidence level. The symbol $<$ (respectively $>$) at position (i, j) means that the system on row i performs worse (respectively better) than the system on column j.

for this task, first-order semi-CRF does not perform significantly better than simple linear-chain CRF. We also observe that SC^3 outperforms C^1, C^2, and SC^1 significantly, while it outperforms C^3 and SC^2 with p-values at most 7.4% and 1.7% respectively. Figure 4 shows the average number of segment label patterns and the average running time of high-order semi-CRFs as a function of the maximum order.

The CRFs in Table 3 do not use begin-inside-outside (BIO) encoding of the labels. In the labeling protocol described above for this problem, although the label O indicates the outside of any argument, we do not differentiate between the beginning and the insides of an argument. In Table 5, we report the F1 scores of C^1, C^2, and C^3 using BIO encoding (C^1-BIO, C^2-BIO, and C^3-BIO respectively). We use *Arg1-B*, *Arg2-B*, and *Arg1Arg2-B* to indicate the beginning of an argument and use *Arg1-I*, *Arg2-I*, and *Arg1Arg2-I* to indicate the insides of an argument. The scores are computed after removing the B and I suffixes in the labels. From the results in Table 5, BIO encoding does not help C^1-BIO and C^2-BIO much, but it helps to improve C^3-BIO substantially. Overall, C^3-BIO achieves the best average F1 score (51.11%) for the relation argument detection problem. Comparing C^3-BIO and SC^3 on each individual relation, we note that SC^3 is useful for *PHYS* where

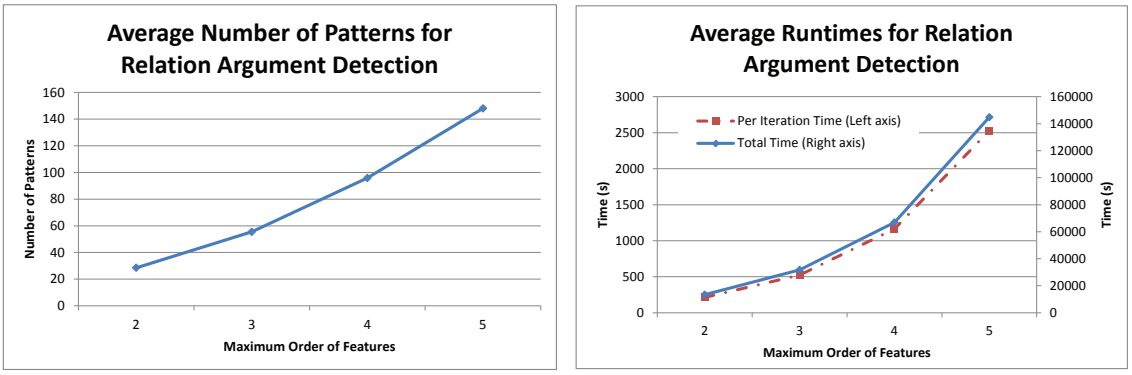


Figure 4: Average number of segment label patterns (left) and average running time (right) of high-order semi-CRFs as a function of maximum order for relation argument detection.

Type	C^1	C^2	C^3	SC^3	C^1-BIO	C^2-BIO	C^3-BIO
Part-Whole	38.68	41.41	46.52	44.30	38.66	41.23	**50.30**
Phys	33.24	33.60	35.20	**42.46**	33.81	34.77	36.88
Org-Aff	60.56	63.28	64.93	64.86	61.33	64.33	**67.50**
Gen-Aff	31.00	35.84	40.16	38.09	30.38	35.03	**43.37**
Per-Soc	53.67	58.62	58.31	57.07	55.07	58.50	**59.37**
Art	40.30	43.80	46.35	48.78	40.62	43.01	**49.25**
Average	42.91	46.09	48.58	49.26	43.31	46.15	**51.11**

Table 5: F1 scores of different (non-semi) CRF taggers for relation argument detection using BIO encoding of the labels (C^1-BIO, C^2-BIO, and C^3-BIO). The scores of C^1, C^2, C^3, and SC^3 are copied from Table 3 for comparison.

the arguments are located further apart. C^3-BIO, on the other hand, is useful for other relations where the arguments are located near to each other.

3.2.2 Punctuation Prediction

In this experiment, we evaluated the performance of high-order semi-CRFs on the punctuation prediction task. This task is usually used as a post-processing step for automatic speech recognition systems to add punctuations to the transcribed conversational speech texts (Liu et al., 2005; Lu and Ng, 2010). Previous evaluations on the IWSLT corpus (Paul, 2009) have shown that capturing long-range dependencies is useful for the task (Lu and Ng, 2010). In the experiment, we used high-order CRFs and high-order semi-CRFs to capture long-range dependencies in the labels and showed that they outperform linear-chain CRF and first-order semi-CRF on movie transcripts data, which contains 5450 conversational speech texts with annotated punctuations from various movie transcripts online. We used

Tag	C^1	C^2	C^3	SC^1	SC^2	SC^3
Comma	59.29	59.70	59.90	**61.13**	60.89	60.35
Period	75.37	75.37	75.46	75.03	**78.97**	78.82
QMark	58.18	59.54	60.57	57.61	**74.05**	73.56
All	66.21	66.53	66.85	66.73	**70.85**	70.47

Table 6: F1 scores for punctuation prediction task. The last row contains the micro-averaged scores.

	C^2	C^3	SC^1	SC^2	SC^3
C^1	0.155<	0.048<	0.043<	**0.001**<	**0.001**<
C^2	–	0.153<	0.289<	**0.001**<	**0.001**<
C^3	–	–	0.378>	**0.001**<	**0.001**<
SC^1	–	–	–	**0.001**<	**0.001**<
SC^2	–	–	–	–	0.044>

Table 7: The values of p' obtained in the statistical significance tests comparing CRFs and semi-CRFs of different orders in the punctuation prediction task, where the p-value of the significance test is smaller than p'. Figures in bold are where the difference is statistically significant at the 1% confidence level. The symbol $<$ (respectively $>$) at position (i, j) means that the system on row i performs worse (respectively better) than the system on column j.

60% of the texts for training and the remaining 40% for testing. The punctuation and case information are removed, and the words are tagged with different labels.

Originally, there are 4 labels: None, Comma, Period, and QMark, which respectively indicate that no punctuation, a comma, a period, or a question mark comes immediately after the current word. To help capture the long-range dependencies, we added 6 more labels: None-Comma, None-Period, None-QMark, Comma-Comma, QMark-QMark, and Period-Period. The left parts of these labels serve the same purpose as the original four labels. The right parts of the labels indicate that the current word is the beginning of a text segment which ends in comma, period, or question mark. This part is used to capture useful information at the beginning of the segment. For example, the sentence "no, she is working." would be labeled as "no/Comma-Comma she/None-Period is/None working/Period". In this case, *she is working* is a text segment (with length 3) that ends with a period. This information is marked in the label of the word *working* and the right part of the label of the word *she*. The text segment *no* (with length 1) is also labeled in a similar way.

We reported the F1 scores of the models in Table 6. We used the combinations of words and their positions relatively to the current position as zeroth-order features. For first-order features, we used transitions without any observation, and transitions with the current or previous words, as well as their combinations. C^k uses k^{th}-order Markov features, while

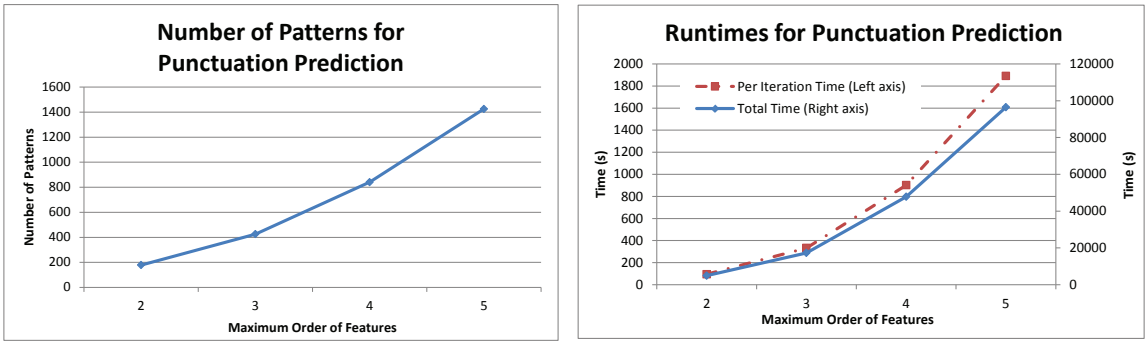


Figure 5: Number of segment label patterns (left) and running time (right) of high-order semi-CRFs as a function of maximum order for the punctuation prediction data set.

SC^k uses k^{th}-order semi-Markov transition features with the observed words in the last segment.

The scores reported in Table 6 are lower than those of the IWSLT corpus (Lu and Ng, 2010) because online movie transcripts are usually annotated by different people, and they tend to put the punctuations slightly differently. Besides, in movies, people sometimes use declarative sentences as questions. Hence, the punctuations are harder to predict. Nevertheless, the results have clearly shown that high-order semi-CRFs can capture long-range dependencies with the help of additional labels and can achieve more than 3.6% improvement in F1 score compared to the CRFs and first-order semi-CRF. SC^k also outperforms C^k for all k. For this task, using third-order semi-Markov features decrease the performance of SC^3 slightly compared to SC^2. From Table 7, we see that the p-value of the statistical significance test comparing SC^2 and SC^3 is at most 4.4%, while both SC^2 and SC^3 significantly outperform the other models. Figure 5 shows the number of segment label patterns and the running time of high-order semi-CRFs as a function of the maximum order.

3.2.3 Bibliography Extraction

In this experiment, we consider the problem of bibliography extraction in scientific papers. For this problem, we need to divide a reference, such as those appearing in the *References* section of this paper, into the following 13 types of segments: Author, Booktitle, Date, Editor, Institution, Journal, Location, Note, Pages, Publisher, Tech, Title, or Volume. The problem can be naturally considered as a sequence labeling problem with the above labels. We evaluated the performance of high-order semi-CRFs and CRFs on the bibliography extraction problem with the Cora Information Extraction data set.[4] In the data set, there are 500 instances of references. We used 300 instances for training and the remaining 200 instances for testing.

We reported in Table 8 the F1 scores of the models. In C^1, zeroth-order features include the surrounding words at each position and letter n-grams, and first-order features include

4. The data set is available at `http://people.cs.umass.edu/~mccallum/data.html`.

Tag	C^1	C^2	C^3	SC^1	SC^2	SC^3
Author	94.21	91.65	93.67	93.97	**94.74**	94.00
Booktitle	73.05	75.00	72.39	75.74	**78.11**	76.47
Date	95.67	**96.68**	94.36	95.19	95.43	95.70
Editor	68.57	**72.73**	66.67	57.14	58.82	54.55
Institution	68.57	64.71	64.71	**70.27**	**70.27**	64.86
Journal	78.08	**78.32**	**78.32**	77.55	77.55	75.68
Location	**70.33**	69.66	68.13	68.13	67.39	65.22
Note	**66.67**	57.14	57.14	57.14	**66.67**	**66.67**
Pages	84.82	**87.83**	85.34	85.96	86.96	87.18
Publisher	84.62	84.62	83.54	84.62	**86.08**	**86.08**
Tech	77.78	**80.00**	**80.00**	77.78	77.78	77.78
Title	89.62	85.42	86.73	90.18	**92.23**	90.95
Volume	66.23	**75.68**	72.60	71.90	72.37	75.00
All	85.34	85.47	84.77	85.67	**86.67**	86.07

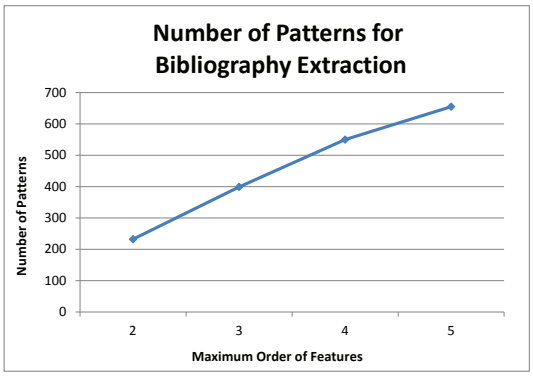
Table 8: F1 scores for bibliography extraction task. The last row contains the micro-averaged scores.

	C^2	C^3	SC^1	SC^2	SC^3
C^1	0.393<	0.174>	0.198<	**0.004**<	0.095<
C^2	–	0.073>	0.351<	0.019<	0.161<
C^3	–	–	0.095<	**0.002**<	0.030<
SC^1	–	–	–	**0.003**<	0.200<
SC^2	–	–	–	–	0.025>

Table 9: The values of p' obtained in the statistical significance tests comparing CRFs and semi-CRFs of different orders in the bibliography extraction task, where the p-value of the significance test is smaller than p'. Figures in bold are where the difference is statistically significant at the 1% confidence level. The symbol $<$ (respectively $>$) at position (i, j) means that the system on row i performs worse (respectively better) than the system on column j.

transitions with words at the current or previous positions. C^k and SC^k $(1 \leq k \leq 3)$ use additional k^{th}-order Markov and semi-Markov transition features.

From Table 8, high-order semi-CRFs perform generally better than high-order CRFs and first-order semi-CRF. SC^2 achieves the best overall performance with 86.67% F1 score. From Table 9, SC^2 outperforms C^2 and SC^3 with a p-value at most 1.9% and 2.5% respectively, while it outperforms other models significantly. Figure 6 shows the number of segment label patterns and the running time of high-order semi-CRFs as a function of the maximum order.



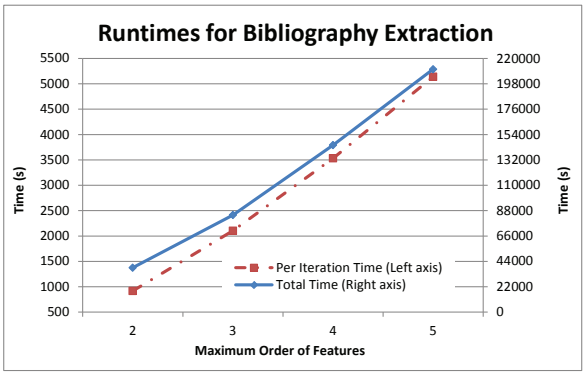


Figure 6: Number of segment label patterns (left) and running time (right) of high-order semi-CRFs as a function of maximum order for the bibliography extraction data set.

3.3 Discussions

From Figures 4, 5, and 6, the number of segment label patterns of high-order features grows about linearly in the maximum order of features. The running time of high-order semi-CRFs on the bibliography extraction task is also nearly linear in the maximum order of the features, while the running times on the relation argument detection task and the punctuation prediction task grow more than linearly in the maximum order of features. We also note that from the time complexity discussions in Section 2.5 and the setup for these experiments, the time complexity of our algorithm is $O(|\mathcal{Z}|^2)$, where $|\mathcal{Z}|$ is the number of segment label patterns.

From Tables 6 and 8, there is a drop in F1 scores for the punctuation prediction task and the bibliography extraction task when we increase the order of the semi-CRFs from 2 to 3. For the punctuation task, the drop is not very significant and the third-order semi-CRF still performs significantly better than the CRFs or the first-order semi-CRFs. For the bibliography extraction task, there is a big drop in the F1 scores for some of the labels and the third-order semi-CRF does not significantly outperform the other models. However, it does not indicate that the third-order semi-CRF is not useful for this task since we fixed the regularization parameter $\sigma_{\text{reg}} = 1$ for all the models in this experiment. If we set $\sigma_{\text{reg}} = 10$ for the third-order semi-CRF, it can achieve 87.45% F1 score and outperform all the other models. In practice, if we have enough data, we can choose a suitable σ_{reg} for each individual model using a validation data set or cross-validation on the training data. We can also allow different regularizers for features of different orders[5] and use a validation set to determine the most suitable combination of regularizers.

An important question in practice is which features (or equivalently, label patterns) should be included in the model. In our experiments, we used all the label patterns that appear in the training data. This simple approach is usually reasonable with a suitable value of the regularization parameter σ_{reg}. For applications where the pattern sparsity assumption is not satisfied, but certain patterns do not appear frequently enough and are

5. This would require a slight change to our regularized log-likelihood function.

not really important, then it is useful to see how we can select a subset of features with few distinct label patterns automatically. One possible approach would be to use boosting type methods (Dietterich et al., 2004) to sequentially select useful features.

For high-order CRFs, it should be possible to use kernels within the approach here. On the handwritten character problem, Taskar et al. (2004) reported substantial improvement in performance with the use of kernels. Use of kernels together with high-order features may lead to further improvements. However, we note that the advantage of the higher order features may become less substantial as the observations become more powerful in distinguishing the classes. Whether the use of higher order features together with kernels brings substantial improvement in performance is likely to be problem dependent.

4. Related Work

A commonly used inference algorithm for CRFs is the clique tree algorithm (Huang and Darwiche, 1996). Defining a feature depending on k (not necessarily consecutive) labels will require forming a clique of size k, resulting in a clique-tree with tree-width greater or equal to k. Inference on such a clique tree will be exponential in k. For sequence models, a feature of order k can be incorporated into a k^{th}-order Markov chain, but the complexity of inference is again exponential in k. Under the label pattern sparsity assumption, our algorithm achieves efficiency by maintaining only information related to a few occurred patterns, while previous algorithms maintain information about all (exponentially many) possible patterns.

Long distance dependencies can also be captured using hierarchical models such as Hierarchical Hidden Markov Model (HHMM) (Fine et al., 1998) or Probabilistic Context Free Grammar (PCFG) (Heemskerk, 1993). The time complexity of inference in an HHMM is $O(\min\{nl^3, n^2l\})$ (Fine et al., 1998; Murphy and Paskin, 2002), where n is the number of states and l is the length of the sequence. Discriminative versions such as hierarchical semi-CRF have also been studied (Truyen et al., 2008). Inference in PCFG and its discriminative version can also be efficiently done in $O(ml^3)$ where m is the number of productions in the grammar (Jelinek et al., 1992). These methods are able to capture dependencies of arbitrary lengths, unlike k^{th}-order Markov chains. However, to do efficient learning with these methods, the hierarchical structure of the examples needs to be provided. For example, if we use PCFG to do character sequence labeling, we need to provide the parse trees for efficient learning; providing the labels for each character is not sufficient. Hence, a training set that has not been labeled with hierarchical labels will need to be relabeled before it can be trained efficiently. Alternatively, methods that employ hidden variables can be used (e.g., to infer the hidden parse tree) but the optimization problem is no longer convex and local optima can sometimes be a problem. The high-order semi-CRF presented in this paper allows us to capture a different class of dependencies that does not depend on hierarchical structures in the data, while keeping the high-order semi-CRF objective a convex optimization problem.

Another work on using high-order features for CRFs was independently done by Qian et al. (2009). Their work applies to a larger class of CRFs, including those requiring exponential time for inference, and they did not identify subclasses for which inference is guaranteed to be efficient. For sequence labeling with high-order features, Qian and Liu

(2012) developed an efficient decoding algorithm under the assumption that all the high-order features have non-negative weights. Their decoding algorithm requires quadratic running time on the number of high-order features in the worst case.

There are other models similar to the high-order CRF with pattern sparsity assumption (Ye et al., 2009), a special case of the high-order semi-CRF presented in this paper. They include the CRFs that use the sparse higher-order potentials (Rother et al., 2009) or the pattern-based potentials (Komodakis and Paragios, 2009). Rother et al. (2009) proposed a method for minimization of sparse higher order energy functions by first transforming them into a quadratic functions and then employing efficient inference algorithms to minimize these resulting functions. For the pattern-based potentials, Komodakis and Paragios (2009) derived an efficient message-passing algorithm for inference. The algorithm is based on the master-slave framework where the original high-order optimization problem is decomposed into smaller subproblems that can be solved easily. Other tractable inference algorithms with high-order potentials include the α-expansion and $\alpha\beta$-swap algorithms for the $\mathcal{P}^n$ Potts model (Kohli et al., 2007) and the MAP message passing algorithm for cardinality and order potentials (Tarlow et al., 2010). A special case of the order potentials, the before-after potential (Tarlow et al., 2010), can also be used to capture some semi-Markov structures in the data labelings.

5. Conclusion

The label pattern sparsity assumption often holds in real applications, and we give efficient inference algorithms for CRFs using high-order dependencies between labels or segments when the pattern sparsity assumption is satisfied. This allows high-order features to be explored in feature engineering for real applications. We studied the conditions that are favorable for using high-order features in CRFs with a synthetic data set, and demonstrated that using simple high-order features can lead to performance improvement on a handwriting recognition problem. We also demonstrated that high-order semi-CRFs outperform high-order CRFs and first-order semi-CRF in segmentation problems like relation argument detection, punctuation prediction, and bibliography extraction.

Acknowledgments

This material is based on research sponsored by DSO under grant DSOCL11102 and by the Air Force Research Laboratory, under agreement number FA2386-09-1-4123. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government. The authors also would like to thank Sumit Bhagwani for his help with the HOSemiCRF package and the anonymous reviewers for their constructive comments.

Appendix A. Correctness of the Forward and Backward Algorithms

In this appendix, we will prove the correctness of the forward and backward algorithms described in Section 2. We shall prove two lemmas and then provide the proofs for the correctness of the forward and backward algorithms as well as the marginal computation.

Lemma 1 below gives the key properties that can be used in an inductive proof. Lemma 1(a) shows that we can partition the segmentations using the forward states. Lemma 1(b-c) show that considering all $(\mathbf{p}^k, y) : \mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{p}^k y$ is sufficient for obtaining the sum over all sequences $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{z}y$, while Lemma 1(d) is used to show that the features are counted correctly.

Lemma 1 *Let* $\mathbf{s}$ *be a segmentation for a prefix of* $\mathbf{x}$. *Let* $\omega(\mathbf{s}, t) = \exp(\sum_{k=1}^m \lambda_k f_k(\mathbf{x}, \mathbf{s}, t))$ *and* $\omega(\mathbf{s}) = \exp(\sum_{k=1}^m \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t)) = \prod_{t=1}^{|\mathbf{s}|} \omega(\mathbf{s}, t)$.
(a) For any segment label sequence $\mathbf{z}$, *there exists a unique* $\mathbf{p}^i \in \mathcal{P}$ *such that* $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{z}$.
(b) For any segment label sequence $\mathbf{z}$ *and* $y \in \mathcal{Y}$, *if* $\mathbf{p}^k \leq^s_{\mathcal{P}} \mathbf{z}$ *and* $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{p}^k y$, *then* $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{z}y$.
(c) For any $\mathbf{z}^a \in \mathcal{Z}$, $y \in \mathcal{Y}$, *and any segment label sequence* $\mathbf{z}$, *if* $\mathbf{z}^a \leq^s \mathbf{z}y$, *and* $\mathbf{p}^k \leq^s_{\mathcal{P}} \mathbf{z}$, *then* $\mathbf{z}^a \leq^s \mathbf{p}^k y$.
(d) Let $\mathbf{s} = ((u_1, v_1, y_1), \ldots, (u_{|\mathbf{s}|}, v_{|\mathbf{s}|}, y_{|\mathbf{s}|}))$ *and let* $\mathbf{p}^{k_t} \leq^s_{\mathcal{P}} y_1 y_2 \ldots y_t$ *for* $t = 1, \ldots, |\mathbf{s}|$. *Then* $\omega(\mathbf{s}) = \prod_{t=1}^{|\mathbf{s}|} \Psi_{\mathbf{x}}(u_t, v_t, \mathbf{p}^{k_{t-1}} y_t) = \omega(\mathbf{s}_{1:|\mathbf{s}|-1}) \Psi_{\mathbf{x}}(u_{|\mathbf{s}|}, v_{|\mathbf{s}|}, \mathbf{p}^{k_{|\mathbf{s}|-1}} y_{|\mathbf{s}|})$.

A.1 Proof of Lemma 1

(a) The intersection of $\mathcal{P}$ and the set of prefixes of $\mathbf{z}$ contains at least one element ϵ, and is finite.

(b) We have $\mathbf{p}^i \leq^s \mathbf{p}^k y \leq^s \mathbf{z}y$. Furthermore, if $\mathbf{p}^j \leq^s \mathbf{z}y$, then we have $\mathbf{p}^j_{1:|\mathbf{p}^j|-1} \leq^s \mathbf{z}$. Thus, $\mathbf{p}^j_{1:|\mathbf{p}^j|-1} \leq^s \mathbf{p}^k$ since $\mathbf{p}^k \leq^s_{\mathcal{P}} \mathbf{z}$. Hence, $\mathbf{p}^j = \mathbf{p}^j_{1:|\mathbf{p}^j|-1} y \leq^s \mathbf{p}^k y$. Since $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{p}^k y$, we have $\mathbf{p}^j \leq^s \mathbf{p}^i$. Therefore, $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{z}y$.

(c) Since $\mathbf{z}^a_{1:|\mathbf{z}^a|-1} \leq^s \mathbf{z}$ and $\mathbf{p}^k \leq^s_{\mathcal{P}} \mathbf{z}$, we have $\mathbf{z}^a_{1:|\mathbf{z}^a|-1} \leq^s \mathbf{p}^k$. Thus, $\mathbf{z}^a \leq^s \mathbf{p}^k y$.

(d) Straightforward from part (c) and definition of $\Psi_{\mathbf{x}}$.

Lemma 2 below serves the same purpose as Lemma 1 for showing correctness.

Lemma 2 *Let* $\mathbf{s}$ *be a segmentation for a suffix of* $\mathbf{x}$. *Let* $\omega(\mathbf{s}, t|\mathbf{s}^i) = \exp(\sum_{k=1}^m \lambda_k f_k(\mathbf{x}, \mathbf{s}, t|\mathbf{s}^i))$ *and* $\omega(\mathbf{s}|\mathbf{s}^i) = \exp(\sum_{k=1}^m \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t|\mathbf{s}^i)) = \prod_{t=1}^{|\mathbf{s}|} \omega(\mathbf{s}, t|\mathbf{s}^i)$.
(a) For all $\mathbf{s}^i \in \mathcal{S}$ *and* $y \in \mathcal{Y}$, *there exists a unique* $\mathbf{s}^k \in \mathcal{S}$ *such that* $\mathbf{s}^k \leq^s_{\mathcal{S}} \mathbf{s}^i y$.
(b) For any $\mathbf{z}^a \in \mathcal{Z}$ *and any segment label sequences* $\mathbf{z}_1, \mathbf{z}_2$, *if* $\mathbf{z}^a \leq^s \mathbf{z}_1\mathbf{z}_2$, *and* $\mathbf{s}^i \leq^s_{\mathcal{S}} \mathbf{z}_1$, *then* $\mathbf{z}^a \leq^s \mathbf{s}^i \mathbf{z}_2$.
(c) If $\mathbf{s}^k \leq^s_{\mathcal{S}} \mathbf{s}^i y$, *and* $(u, v, y) \cdot \mathbf{s}$ *is a segmentation for* $\mathbf{x}_{u:|\mathbf{x}|}$, *then* $\omega((u, v, y) \cdot \mathbf{s}|\mathbf{s}^i) = \Psi_{\mathbf{x}}(u, v, \mathbf{s}^i y)\omega(\mathbf{s}|\mathbf{s}^k)$.

A.2 Proof of Lemma 2

(a) Note that $y \in \mathcal{S}$ and $y \leq^s \mathbf{s}^i y$ and the number of suffixes of $\mathbf{s}^i y$ is finite.

(b) This is clearly true if $\mathbf{z}^a$ is not longer than $\mathbf{z}_2$. If $\mathbf{z}^a$ is longer than $\mathbf{z}_2$, let $\mathbf{p}$ be the prefix of $\mathbf{z}^a$ obtained by stripping off the suffix $\mathbf{z}_2$. Then $\mathbf{p}$ is a suffix of $\mathbf{z}_1$ and $\mathbf{p} \in \mathcal{S}$. Since $\mathbf{s}^i$ is the longest suffix of $\mathbf{z}_1$ in $\mathcal{S}$, $\mathbf{p}$ is a suffix of $\mathbf{s}^i$, thus $\mathbf{z}^a = \mathbf{p}\mathbf{z}_2$ is a suffix of $\mathbf{s}^i\mathbf{z}_2$.

(c) From part (b), we have $\omega(\mathbf{s}|\mathbf{s}^i y) = \omega(\mathbf{s}|\mathbf{s}^k)$. Thus, $\omega((u,v,y)\cdot\mathbf{s}|\mathbf{s}^i) = \Psi_{\mathbf{x}}(u,v,\mathbf{s}^i y)\omega(\mathbf{s}|\mathbf{s}^i y) = \Psi_{\mathbf{x}}(u,v,\mathbf{s}^i y)\omega(\mathbf{s}|\mathbf{s}^k)$.

A.3 Correctness of the Forward Algorithm

Given the forward variables $\alpha_{\mathbf{x}}(j,\mathbf{p}^i)$ as defined in Section 2

$$\alpha_{\mathbf{x}}(j,\mathbf{p}^i) = \sum_{\mathbf{s}\in\mathbf{p}_{j,\mathbf{p}^i}} \exp(\sum_{k=1}^{m}\sum_{t=1}^{|\mathbf{s}|}\lambda_k f_k(\mathbf{x},\mathbf{s},t)) = \sum_{\mathbf{s}\in\mathbf{p}_{j,\mathbf{p}^i}} \omega(\mathbf{s}),$$

we prove that the following recurrence can be used to compute $\alpha_{\mathbf{x}}(j,\mathbf{p}^i)$'s by induction on j,

$$\alpha_{\mathbf{x}}(j,\mathbf{p}^i) = \sum_{d=0}^{L-1}\sum_{(\mathbf{p}^k,y):\mathbf{p}^i\le_{\mathcal{P}}^{s}\mathbf{p}^k y} \Psi_{\mathbf{x}}(j-d,j,\mathbf{p}^k y)\alpha_{\mathbf{x}}(j-d-1,\mathbf{p}^k). \quad (2)$$

Base case: If $j=1$, for any $\mathbf{p}^i \in \mathcal{P}$, we can initialize the values of $\alpha_{\mathbf{x}}(1,\mathbf{p}^i)$ such that

$$\alpha_{\mathbf{x}}(1,\mathbf{p}^i) = \sum_{\mathbf{s}\in\mathbf{p}_{1,\mathbf{p}^i}} \exp(\sum_{k=1}^{m}\sum_{t=1}^{|\mathbf{s}|}\lambda_k f_k(\mathbf{x},\mathbf{s},t)) = \sum_{\mathbf{s}\in\mathbf{p}_{1,\mathbf{p}^i}} \omega(\mathbf{s}).$$

Inductive step: Assume that for all $j' < j$ and $\mathbf{p}^i \in \mathcal{P}$, we have

$$\alpha_{\mathbf{x}}(j',\mathbf{p}^i) = \sum_{\mathbf{s}\in\mathbf{p}_{j',\mathbf{p}^i}} \exp(\sum_{k=1}^{m}\sum_{t=1}^{|\mathbf{s}|}\lambda_k f_k(\mathbf{x},\mathbf{s},t)) = \sum_{\mathbf{s}\in\mathbf{p}_{j',\mathbf{p}^i}} \omega(\mathbf{s}).$$

Then, using Lemma 1,

$$\begin{aligned}
\alpha_{\mathbf{x}}(j,\mathbf{p}^i) &= \textstyle\sum_{\mathbf{s}\in\mathbf{p}_{j,\mathbf{p}^i}} \omega(\mathbf{s}) \\
&= \textstyle\sum_{d=0}^{L-1}\sum_{(\mathbf{p}^k,y):\mathbf{p}^i\le_{\mathcal{P}}^{s}\mathbf{p}^k y}\sum_{\mathbf{s}\in\mathbf{p}_{j-d-1,\mathbf{p}^k}} \omega(\mathbf{s}\cdot(j-d,j,y)) \\
&= \textstyle\sum_{d=0}^{L-1}\sum_{(\mathbf{p}^k,y):\mathbf{p}^i\le_{\mathcal{P}}^{s}\mathbf{p}^k y}\sum_{\mathbf{s}\in\mathbf{p}_{j-d-1,\mathbf{p}^k}} [\Psi_{\mathbf{x}}(j-d,j,\mathbf{p}^k y)\textstyle\prod_{t=1}^{|\mathbf{s}|}\omega(\mathbf{s},t)] \\
&= \textstyle\sum_{d=0}^{L-1}\sum_{(\mathbf{p}^k,y):\mathbf{p}^i\le_{\mathcal{P}}^{s}\mathbf{p}^k y}\Psi_{\mathbf{x}}(j-d,j,\mathbf{p}^k y)\sum_{\mathbf{s}\in\mathbf{p}_{j-d-1,\mathbf{p}^k}}\textstyle\prod_{t=1}^{|\mathbf{s}|}\omega(\mathbf{s},t) \\
&= \textstyle\sum_{d=0}^{L-1}\sum_{(\mathbf{p}^k,y):\mathbf{p}^i\le_{\mathcal{P}}^{s}\mathbf{p}^k y}\Psi_{\mathbf{x}}(j-d,j,\mathbf{p}^k y)\alpha_{\mathbf{x}}(j-d-1,\mathbf{p}^k).
\end{aligned}$$

Hence, by induction, Recurrence (2) correctly computes the forward variables $\alpha_{\mathbf{x}}(j,\mathbf{p}^i)$'s.

A.4 Correctness of the Backward Algorithm

Given the backward variables $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$ as defined in Section 2

$$\beta_{\mathbf{x}}(j, \mathbf{s}^i) = \sum_{\mathbf{s} \in \mathbf{s}_j} \exp(\sum_{k=1}^{m} \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t | \mathbf{s}^i)) = \sum_{\mathbf{s} \in \mathbf{s}_j} \omega(\mathbf{s}|\mathbf{s}^i),$$

we prove that the following recurrence can be used to compute $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$'s by induction on j,

$$\beta_{\mathbf{x}}(j, \mathbf{s}^i) = \sum_{d=0}^{L-1} \sum_{(\mathbf{s}^k, y): \mathbf{s}^k \leq_{\mathcal{S}}^{s} \mathbf{s}^i y} \Psi_{\mathbf{x}}(j, j+d, \mathbf{s}^i y) \beta_{\mathbf{x}}(j+d+1, \mathbf{s}^k). \tag{3}$$

Base case: If $j = |\mathbf{x}|$, for any $\mathbf{s}^i \in \mathcal{S}$, we can initialize the values of $\beta_{\mathbf{x}}(|\mathbf{x}|, \mathbf{s}^i)$ such that

$$\beta_{\mathbf{x}}(|\mathbf{x}|, \mathbf{s}^i) = \sum_{\mathbf{s} \in \mathbf{s}_{|\mathbf{x}|}} \exp(\sum_{k=1}^{m} \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t | \mathbf{s}^i)) = \sum_{\mathbf{s} \in \mathbf{s}_{|\mathbf{x}|}} \omega(\mathbf{s}|\mathbf{s}^i).$$

Inductive step: Assume that for all $j' > j$ and $\mathbf{s}^i \in \mathcal{S}$, we have

$$\beta_{\mathbf{x}}(j', \mathbf{s}^i) = \sum_{\mathbf{s} \in \mathbf{s}_{j'}} \exp(\sum_{k=1}^{m} \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t | \mathbf{s}^i)) = \sum_{\mathbf{s} \in \mathbf{s}_{j'}} \omega(\mathbf{s}|\mathbf{s}^i).$$

Then, using Lemma 2,

$$\begin{aligned}
\beta_{\mathbf{x}}(j, \mathbf{s}^i) &= \textstyle\sum_{\mathbf{s} \in \mathbf{s}_j} \omega(\mathbf{s}|\mathbf{s}^i) \\
&= \textstyle\sum_{d=0}^{L-1} \sum_{(\mathbf{s}^k, y): \mathbf{s}^k \leq_{\mathcal{S}}^{s} \mathbf{s}^i y} \sum_{\mathbf{s} \in \mathbf{s}_{j+d+1}} \omega((j, j+d, y) \cdot \mathbf{s}|\mathbf{s}^i) \\
&= \textstyle\sum_{d=0}^{L-1} \sum_{(\mathbf{s}^k, y): \mathbf{s}^k \leq_{\mathcal{S}}^{s} \mathbf{s}^i y} \sum_{\mathbf{s} \in \mathbf{s}_{j+d+1}} \Psi_{\mathbf{x}}(j, j+d, \mathbf{s}^i y) \omega(\mathbf{s}|\mathbf{s}^k) \\
&= \textstyle\sum_{d=0}^{L-1} \sum_{(\mathbf{s}^k, y): \mathbf{s}^k \leq_{\mathcal{S}}^{s} \mathbf{s}^i y} \Psi_{\mathbf{x}}(j, j+d, \mathbf{s}^i y) \beta_{\mathbf{x}}(j+d+1, \mathbf{s}^k).
\end{aligned}$$

Hence, by induction, Recurrence (3) correctly computes the backward variables $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$'s.

A.5 Correctness of the Marginal Computation

Consider a segmentation $\mathbf{s}$ such that the segment label sequence of $\mathbf{s}$ contains $\mathbf{z}$ as a subsequence with the last segment of $\mathbf{z}$ having boundaries (u, v). Suppose $\mathbf{s} = \mathbf{s}_1 \cdot (u, v, y) \cdot \mathbf{s}_2$ and let $\mathbf{y}_1$ be the segment label sequence of $\mathbf{s}_1$. If $\mathbf{p}^i \leq_{\mathcal{P}}^{s} \mathbf{y}_1$, then we have $\mathbf{p}^i y \leq_{\mathcal{S}}^{s} \mathbf{y}_1 y$. In this case, it can be verified that $\omega(\mathbf{s}) = \omega(\mathbf{s}_1) \Psi(u, v, \mathbf{p}^i y) \omega(\mathbf{s}_2 | \mathbf{p}^i y)$. The marginal formula thus follows easily.

Appendix B. An Example for the Algorithms

In this appendix, we give an example to illustrate our algorithms. For simplicity, we use the second-order CRF as our model. Extensions to higher-order CRFs or semi-CRFs should be straightforward by respectively expanding the set of segment label patterns or summing over all the possible lengths d of the segments.

i	$f_i(\mathbf{x}, \mathbf{s}, t)$
1	$x_t = Peter \wedge s_t = P$
2	$x_t = goes \wedge s_t = O$
3	$x_t = to \wedge s_t = O$
4	$x_t = Britain \wedge s_t = L$
5	$x_t = and \wedge s_t = O$
6	$x_t = France \wedge s_t = L$
7	$x_t = annually \wedge s_t = O$
8	$x_t = . \wedge s_t = O$
9	$s_{t-2}s_{t-1}s_t = LOL$

Table 10: List of features for the example in Appendix B.

$t\backslash\mathbf{z}$	P	O	L	LOL
1	1	0	0	0
2	0	1	0	0
3	0	1	0	0
4	0	0	1	0
5	0	1	0	0
6	0	0	1	1
7	0	1	0	0
8	0	1	0	0

Table 11: The values of $\sum_{i:\mathbf{z}^i=\mathbf{z}} \lambda_i g_i(\mathbf{x}, u_t, v_t) = \sum_{i:\mathbf{z}^i=\mathbf{z}} \lambda_i g_i(\mathbf{x}, t, t)$.

In this example, let $\mathbf{x}$ be the sentence "Peter goes to Britain and France annually.". Assume there are 9 binary features defined by Boolean predicates as in Table 10, and each $\lambda_i = 1$. The label set is $\{P, O, L\}$ where P represents *Person*, L represents *Location* and O represents *Others*. Note that for second-order CRFs, the length of all the segments is 1 and thus $s_t = y_t$ for all t.

The segment label pattern set is $\mathcal{Z} = \{P, O, L, LOL\}$. Table 11 shows the sum of the weights for features with the same segment label pattern at each position. We have $\mathcal{P} = \{\epsilon, P, O, L, LO\}$ and $\mathcal{S} = \{P, O, L, PP, PO, PL, OP, OO, OL, LP, LO, LL, LOP, LOO, LOL\}$. The tables for $\ln \alpha_{\mathbf{x}}$ and $\ln \beta_{\mathbf{x}}$ are shown in Table 12 and Table 13 respectively.

In Figure 7, we give a diagram to show the messages passed from step $j-1$ to step j to compute the forward variables $\alpha_{\mathbf{x}}$. We also give a diagram in Figure 8 to show some messages passed from step $j+1$ to step j to compute the backward variables $\beta_{\mathbf{x}}$.

We illustrate the computation of $\alpha_{\mathbf{x}}$ with $\alpha_{\mathbf{x}}(6, L)$. The condition $(\mathbf{p}^k, y) : \mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{p}^k y$ with $\mathbf{p}^i = L$ gives us the following 5 pairs as $(\mathbf{p}^k, y)$: $\{(\epsilon, L), (P, L), (O, L), (L, L), (LO, L)\}$. Thus,

$$\begin{aligned}
\alpha_{\mathbf{x}}(6, L) &= \alpha_{\mathbf{x}}(5, \epsilon)\Psi_{\mathbf{x}}(6, 6, L) + \alpha_{\mathbf{x}}(5, P)\Psi_{\mathbf{x}}(6, 6, PL) + \alpha_{\mathbf{x}}(5, O)\Psi_{\mathbf{x}}(6, 6, OL) + \\
&\quad \alpha_{\mathbf{x}}(5, L)\Psi_{\mathbf{x}}(6, 6, LL) + \alpha_{\mathbf{x}}(5, LO)\Psi_{\mathbf{x}}(6, 6, LOL) \\
&= 0\Psi_{\mathbf{x}}(6, 6, L) + \alpha_{\mathbf{x}}(5, P)e + \alpha_{\mathbf{x}}(5, O)e + \alpha_{\mathbf{x}}(5, L)e + \alpha_{\mathbf{x}}(5, LO)e^2.
\end{aligned}$$

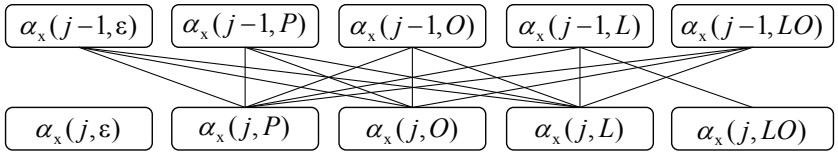


Figure 7: Messages passed from step $j-1$ to step j in order to compute the forward variables. For example, $\alpha_{\mathbf{x}}(j,O)$ is computed from $\alpha_{\mathbf{x}}(j-1,\epsilon)$, $\alpha_{\mathbf{x}}(j-1,P)$, $\alpha_{\mathbf{x}}(j-1,O)$, and $\alpha_{\mathbf{x}}(j-1,LO)$.

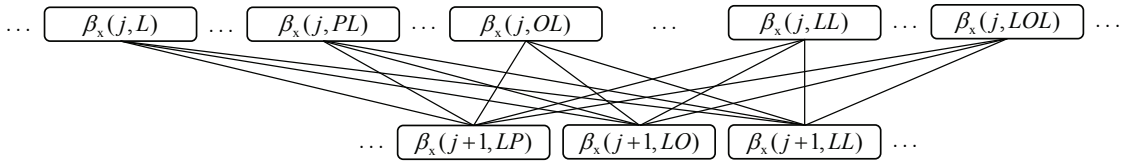


Figure 8: Some messages passed from step $j+1$ to step j in order to compute the backward variables. In this example, all the variables $\beta_{\mathbf{x}}(j,L)$, $\beta_{\mathbf{x}}(j,PL)$, $\beta_{\mathbf{x}}(j,OL)$, $\beta_{\mathbf{x}}(j,LL)$, and $\beta_{\mathbf{x}}(j,LOL)$ are computed from $\beta_{\mathbf{x}}(j+1,LP)$, $\beta_{\mathbf{x}}(j+1,LO)$, and $\beta_{\mathbf{x}}(j+1,LL)$.

$j\backslash\mathbf{p}^i$	ϵ	P	O	L	LO
1	-∞	1.00	0.00	0.00	-∞
2	-∞	1.55	2.31	1.55	1.00
3	-∞	3.10	3.87	3.12	2.55
4	-∞	4.65	4.42	5.65	3.10
5	-∞	6.21	6.35	6.21	6.65
6	-∞	7.76	7.52	9.21	6.21
7	-∞	9.60	9.45	9.59	10.21
8	-∞	11.14	11.91	11.14	10.59

Table 12: The values of $\ln \alpha_{\mathbf{x}}(j,\mathbf{p}^i)$.

We also have $Z_{\mathbf{x}} = \alpha_{\mathbf{x}}(8,\epsilon) + \alpha_{\mathbf{x}}(8,P) + \alpha_{\mathbf{x}}(8,O) + \alpha_{\mathbf{x}}(8,L) + \alpha_{\mathbf{x}}(8,LO) = e^{12.696}$.

We now illustrate the computation of $\beta_{\mathbf{x}}$ with $\beta_{\mathbf{x}}(5,OL)$. The condition $(\mathbf{s}^k,y) : \mathbf{s}^k \leq^s_{\mathcal{S}} \mathbf{s}^i y$ with $\mathbf{s}^i = OL$ gives us the following 3 pairs as $(\mathbf{s}^k,y)$: $\{(LP,P),(LO,O),(LL,L)\}$. Thus,

$$\begin{aligned}\beta_{\mathbf{x}}(5,OL) &= \beta_{\mathbf{x}}(6,LP)\Psi_{\mathbf{x}}(5,5,OLP) + \beta_{\mathbf{x}}(6,LO)\Psi_{\mathbf{x}}(5,5,OLO) + \\ &\quad \beta_{\mathbf{x}}(6,LL)\Psi_{\mathbf{x}}(5,5,OLL) \\ &= \beta_{\mathbf{x}}(6,LP)e^0 + \beta_{\mathbf{x}}(6,LO)e + \beta_{\mathbf{x}}(6,LL)e^0.\end{aligned}$$

The values of the marginals $P(j,j,\mathbf{z}|\mathbf{x})$ are shown in Table 14. We illustrate the computation of $P(6,6,LOL|\mathbf{x})$ from the forward and backward variables. The condition

$j\backslash\mathbf{s}^i$	P	O	L	PP	PO	PL	OP	OO	OL	LP	LO	LL	LOP	LOO	LOL
1	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70
2	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14
3	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59
4	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04
5	6.21	6.21	6.66	6.21	6.21	6.66	6.21	6.21	6.66	6.21	6.21	6.66	6.21	6.21	6.66
6	4.65	4.65	4.65	4.65	4.65	4.65	4.65	4.65	4.65	4.65	5.34	4.65	4.65	4.65	4.65
7	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10
8	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55

Table 13: The values of $\ln \beta_{\mathbf{x}}(j, \mathbf{s}^i)$.

$(\mathbf{p}^i, y) : \mathbf{z} \leq^s \mathbf{p}^i y$ with $\mathbf{z} = LOL$ gives us the only pair (LO, L) as $(\mathbf{p}^i, y)$. Hence,

$$\begin{aligned} P(6, 6, LOL|\mathbf{x}) &= \frac{\alpha_{\mathbf{x}}(5, LO)\beta_{\mathbf{x}}(7, LOL)\Psi_{\mathbf{x}}(6, 6, LOL)}{Z_{\mathbf{x}}} \\ &= \frac{\alpha_{\mathbf{x}}(5, LO)\beta_{\mathbf{x}}(7, LOL)e^2}{Z_{\mathbf{x}}}. \end{aligned}$$

$j\backslash\mathbf{z}$	P	O	L	LOL
1	0.58	0.21	0.21	0.00
2	0.21	0.58	0.21	0.00
3	0.21	0.58	0.21	0.03
4	0.16	0.16	0.68	0.08
5	0.16	0.68	0.16	0.01
6	0.16	0.16	0.68	0.39
7	0.21	0.58	0.21	0.01
8	0.21	0.58	0.21	0.08

Table 14: The marginals $P(j, j, \mathbf{z}|\mathbf{x})$.

References

Aron Culotta, David Kulp, and Andrew McCallum. Gene prediction with conditional random fields. Technical Report UM-CS-2005-028, University of Massachusetts, Amherst, 2005.

Thomas G. Dietterich, Adam Ashenfelter, and Yaroslav Bulatov. Training conditional random fields via gradient tree boosting. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Richard Durbin. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical hidden Markov model: analysis and applications. *Machine Learning*, 32(1):41–62, 1998.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 427–434, 2005.

Josée S. Heemskerk. A probabilistic context-free grammar for disambiguation in morphological parsing. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, pages 183–192, 1993.

Cecil Huang and Adnan Darwiche. Inference in belief networks: a procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

Sorin Istrail. Statistical mechanics, three-dimensionality and NP-completeness: I. Universality of intractability for the partition function of the Ising model across non-planar lattices. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 87–96, 2000.

Frederick Jelinek, John D. Lafferty, and Robert L. Mercer. Basic methods of probabilistic context free grammars. In *Speech Recognition and Understanding. Recent Advances, Trends, and Applications*. Springer, 1992.

Robert H. Kassel. *A Comparison of Approaches to On-line Handwritten Character Recognition*. PhD thesis, Massachusetts Institute of Technology, 1995.

Pushmeet Kohli, M. Pawan Kumar, and Philip H. S. Torr. P3 & beyond: solving energies with higher order cliques. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

Nikos Komodakis and Nikos Paragios. Beyond pairwise energies: efficient optimization for higher-order MRFs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2985–2992, 2009.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. Using conditional random fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 451–458, 2005.

Wei Lu and Hwee Tou Ng. Better punctuation prediction with dynamic conditional random fields. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 177–186, 2010.

Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the 7th Conference on Computational Natural Language Learning*, pages 188–191, 2003.

Kevin P. Murphy and Mark A. Paskin. Linear-time inference in hierarchical HMMs. In *Advances in Neural Information Processing Systems*, pages 833–840, 2002.

Viet Cuong Nguyen, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. Semi-Markov conditional random field with high-order features. In *ICML Workshop on Structured Sparsity: Learning and Inference*, 2011.

Eric W. Noreen. *Computer Intensive Methods for Testing Hypotheses: An Introduction.* Wiley, 1989.

Michael Paul. Overview of the IWSLT 2009 evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 3–27, 2009.

Xian Qian and Yang Liu. Sequence labeling with non-negative weighted higher order features. In *Proceedings of the 26th Conference on Artificial Intelligence*, 2012.

Xian Qian, Xiaoqian Jiang, Qi Zhang, Xuanjing Huang, and Lide Wu. Sparse higher order conditional random fields for improved sequence labeling. In *Proceedings of the 26th International Conference on Machine Learning*, pages 849–856, 2009.

Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Carsten Rother, Pushmeet Kohli, Wei Feng, and Jiaya Jia. Minimizing sparse higher order energy functions of discrete variables. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1382–1389, 2009.

Sunita Sarawagi and William W. Cohen. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, pages 1185–1192, 2004.

Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 134–141, 2003.

Daniel Tarlow, Inmar E. Givoni, and Richard S. Zemel. HOP-MAP: efficient message passing with high order potentials. In *International Conference on Artificial Intelligence and Statistics*, pages 812–819, 2010.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, 2004.

Tran T. Truyen, Dinh Q. Phung, Hung H. Bui, and Svetha Venkatesh. Hierarchical semi-Markov conditional random fields for recursive sequential data. In *Advances in Neural Information Processing Systems*, pages 1657–1664, 2008.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. ACE 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 2006.

Nan Ye, Wee Sun Lee, Hai Leong Chieu, and Dan Wu. Conditional random fields with high-order features for sequence labeling. In *Advances in Neural Information Processing Systems*, pages 2196–2204, 2009.

Alexander Yeh. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 947–953, 2000.

Min Zhang, Jie Zhang, and Jian Su. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 288–295, 2006.