Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data

Charles Sutton  
*University of Massachusetts - Amherst*

Khashayar Rohanimanesh  
*University of Massachusetts - Amherst*

Andrew McCallum  
*University of Massachusetts - Amherst*

Follow this and additional works at: [https://scholarworks.umass.edu/cs_faculty_pubs](https://scholarworks.umass.edu/cs_faculty_pubs)

Recommended Citation

Sutton, Charles; Rohanimanesh, Khashayar; and McCallum, Andrew, "Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data" (2004). *Computer Science Department Faculty Publication Series*. 63. Retrieved from [https://scholarworks.umass.edu/cs_faculty_pubs/63](https://scholarworks.umass.edu/cs_faculty_pubs/63)

This Article is brought to you for free and open access by the Computer Science at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Computer Science Department Faculty Publication Series by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.
Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data

Charles Sutton
Khashayar Rohanimanesh
Andrew McCallum

Department of Computer Science, University of Massachusetts, Amherst, MA 01003

Abstract

In sequence modeling, we often wish to represent complex interaction between labels, such as when performing multiple, cascaded labeling tasks on the same sequence, or when long-range dependencies exist. We present dynamic conditional random fields (DCRFs), a generalization of linear-chain conditional random fields (CRFs) in which each time slice contains a set of state variables and edges—a distributed state representation as in dynamic Bayesian networks (DBNs)—and parameters are tied across slices. Since exact inference can be intractable in such models, we perform approximate inference using several schedules for belief propagation, including tree-based reparameterization (TRP). On a natural-language chunking task, we show that a DCRF performs better than a series of linear-chain CRFs, achieving comparable performance using only half the training data.

1. Introduction

The problem of labeling and segmenting sequences of observations arises in many different areas, including bioinformatics, music modeling, computational linguistics, speech recognition, and information extraction. Dynamic Bayesian networks (DBNs) (Dean & Kanazawa, 1989; Murphy, 2002) are a popular method for probabilistic sequence modeling, because they exploit structure in the problem to compactly represent distributions over multiple state variables. Hidden Markov models (HMMs), an important special case of DBNs, are a classical method for speech recognition (Rabiner, 1989) and part-of-speech tagging (Manning & Schütze, 1999). More complex DBNs have been used for applications as diverse as robot navigation (Theocharou et al., 2001), audio-visual speech recognition (Nefian et al., 2002), activity recognition (Bui et al., 2002), and information extraction (Skounakis et al., 2003; Peshkin & Pfeffer, 2003).

DBNs are typically trained to maximize the joint probability \( p(y, x) \) of a set of observation sequences \( x \) and labels \( y \). However, when the task does not require being able to generate \( x \), such as in segmenting and labeling, modeling the joint distribution is a waste of modeling effort. Furthermore, generative models often must make problematic independence assumptions among the observed nodes in order to achieve tractability. In modeling natural language, for example, we may wish to use features of a word such as its identity, capitalization, prefixes and suffixes, neighboring words, membership in domain-specific lexicons, and category in semantic databases like WordNet—features which have complex interdependencies. Generative models that represent these interdependencies are in general intractable; but omitting such features or modeling them as independent has been shown to hurt accuracy (McCallum et al., 2000).

A solution to this problem is to model instead the conditional probability distribution \( p(y|x) \). The random vector \( x \) can include arbitrary, non-independent, domain-specific feature variables. Because the model is conditional, the dependencies among the features in \( x \) do not need to be explicitly represented. Conditionally-trained models have been shown to perform better than generatively-trained models on many tasks, including document classification (Taskar et al., 2002), part-of-speech tagging (Ratnaparkhi, 1996), extraction of data from tables (Pinto et al., 2003), segmentation of FAQ lists (McCallum et al., 2000), and noun-phrase segmentation (Sha & Pereira, 2003).

Conditional random fields (CRFs) (Lafferty et al., 2001) are undirected graphical models that are conditionally trained. Previous work on CRFs has focused on the linear-chain structure, depicted in Figure 1, in which a first-order Markov assumption is made among labels. This model structure is analogous to conditionally-trained HMMs, and has efficient exact inference algorithms. Often, however,
we wish to represent more complex interaction between labels—for example, when longer-range dependencies exist between labels, when the state can be naturally represented as a vector of variables, or when performing multiple cascaded labeling tasks on the same input sequence (which is prevalent in natural language processing, such as part-of-speech tagging followed by noun-phrase segmentation).

In this paper, we introduce Dynamic CRFs (DCRFs), which are a generalization of linear-chain CRFs that repeat structure and parameters over a sequence of state vectors—allowing us to represent distributed hidden state and complex interaction among labels, as in DBNs, and to use rich, overlapping feature sets, as in conditional models. For example, the factorial structure in Figure 1(b) includes links between cotemporal labels, explicitly modeling limited probabilistic dependencies between different two label sequences. Other types of DCRFs can model higher-order Markov dependence between labels (Figure 2), or incorporate a fixed-size memory. For example, a DCRF for part-of-speech tagging could include for each word a hidden state that is true if any previous word has been tagged as a verb.

Any DCRF with multiple state variables can be collapsed into a linear-chain CRF whose state space is the cross-product of the outcomes of the original state variables. However, such a linear-chain CRF needs exponentially many parameters in the number of variables. Like DBNs, DCRFs represent the joint distribution with fewer parameters by exploiting conditional independence relations.

Within natural-language processing, DCRFs are especially attractive because they are a probabilistic generalization of cascaded, weighted finite-state transducers (Mohri et al., 2002). In general, many sequence-processing problems are traditionally solved by chaining errorful subtasks such as FSTs. In such an approach, however, errors early in processing nearly always cascade through the chain, causing errors in the final output. This problem can be solved by jointly representing the subtasks in a single graphical model, both explicitly representing their dependence, and preserving uncertainty between them. DCRFs can represent dependence between subtasks solved using finite-state transducers, such as phonological and morphological analysis, POS tagging, shallow parsing, and information extraction.

We evaluate DCRFs on a natural-language processing task. A factorial CRF that learns to jointly predict parts of speech and segment noun phrases performs better than cascaded models that perform the two tasks in sequence. Also, we compare several schedules for belief propagation on this task, showing that although exact inference is feasible, approximate inference has lower total training time with no loss in performance.

The rest of the paper is structured as follows. In section 2, we describe the general framework of CRFs. Then, in section 3, we define DCRFs, and explain methods for approximate inference and parameter estimation. In section 4, we present the experimental results. We conclude in section 5.

2. CRFs

Conditional random fields (CRFs) (Lafferty et al., 2001) are undirected graphical models that encode a conditional probability distribution using a given set of features. CRFs are defined as follows. Let $\mathcal{G}$ be an undirected model over sets of random variables $y$ and $x$. As a typical special case, $y = \{y_t\}$ and $x = \{x_t\}$ for $t = 1, \ldots, T$, so that $y$ is a labeling of an observed sequence $x$. If $C = \{(y_c, x_c)\}$ is the set of cliques in $\mathcal{G}$, then CRFs define the conditional probability of a state sequence given the observed sequence as:

$$p_\Lambda(y|x) = \frac{1}{Z(x)} \prod_{c \in C} \Phi(y_c, x_c),$$

where $\Phi$ is a potential function and the partition function $Z(x) = \sum_y \prod_{c \in C} \Phi(y_c, x_c)$ is a normalization factor over all state sequences for the sequence $x$. We assume the potentials factorize according to a set of features $\{f_k\}$, which are given and fixed, so that

$$\Phi(y_c, x_c) = \exp \left( \sum_k \lambda_k f_k(y_c, x_c) \right)$$

The model parameters are a set of real weights $\Lambda = \{\lambda_k\}$, one weight for each feature.

Previous applications use the linear-chain CRF, in which a first-order Markov assumption is made on the hidden variables. A graphical model for this is shown in Figure 1. In this case, the cliques of the conditional model are the nodes and edges, so that there are feature functions $f_k(y_{t-1}, y_t, x, t)$ for each label transition. (Here we write the feature functions as potentially depending on the entire input sequence.) Feature functions can be arbitrary. For example, a feature function $f_k(y_{t-1}, y_t, x, t)$ could be a binary test that has value 1 if and only if $y_{t-1}$ has the label “adjective”, $y_t$ has the label “proper noun”, and $x_t$ begins with a capital letter.
3. Dynamic CRFs

3.1. Model Representation

A Dynamic CRF is a conditionally-trained undirected graphical model whose structure and parameters are repeated over a sequence. As with a DBN, a DCRF can be specified by a template that gives the graphical structure, features, and weights for two time steps, which can then be unrolled given an instance \( x \). The same set of features and weights is used at each sequence position, so that the parameters are tied across the network. Several example templates are given in Figure 2.

Now we give a formal description of the unrolling process. Let \( y = \{y_1 \ldots y_T\} \) be a sequence of random vectors \( y_i = (y_{i1} \ldots y_{im}) \). To give the likelihood equation for arbitrary DCRFs, we require a way to describe a clique in the unrolled graph independent of its position in the sequence. For this purpose we introduce the concept of a clique index. Given a time \( t \), we can denote any variable \( y_{ij} \) in \( y \) by two integers: its index \( j \) in the state vector \( y_t \), and its time offset \( \Delta t = i - t \). We will call a set \( c = \{ (\Delta t, j) \} \) of such pairs a clique index, which denotes a set of variables \( y_{t,c} \) by \( y_{t,c} = \{ y_{i + \Delta t, j} | (\Delta t, j) \in c \} \). That is, \( y_{t,c} \) is the set of variables in the unrolled version of clique index \( c \) at time \( t \).

Now we can formally define DCRFs:

**Definition** Let \( C \) be a set of clique indices, \( F = \{ f_k(y_{t,c}, x, t) \} \) be a set of feature functions and \( \Lambda = \{ \lambda_k \} \) be a set of real-valued weights. Then \( (C, F, \Lambda) \) is a DCRF if and only if

\[
p(y|x) = \frac{1}{Z(x)} \prod_{c \in C} \exp \left( \sum_k \lambda_k f_k(y_{t,c}, x, t) \right)
\]

(3)

where \( Z(x) = \sum_y \prod_{t} \prod_{c \in C} \exp \left( \sum_k \lambda_k f_k(y_{t,c}, x, t) \right) \) is the partition function.

Although we define a DCRF has having the same set of features for all the cliques, in practice, we choose feature functions \( f_k \) so that they are non-zero except on cliques with some index \( c_k \). Thus, we will sometimes think of each clique index having its own set of features and weights, and speak of \( f_k \) and \( \lambda_k \) as having an associated clique index \( c_k \).

DCRFs generalize not only linear-chain CRFs, but more complicated structures as well. For example, in this paper, we use a factorial CRF (FCRF), which has linear chains of labels, with connections between cotemporal labels. We name these after factorial HMMs (Ghahramani & Jordan, 1997). Figure 1(b) shows an unrolled factorial CRF. Consider an CRF with \( L \) chains, where \( y_{\ell,t} \) is the variable in chain \( \ell \) at time \( t \). The clique indices for this DCRF are of the form \( \{0, \ell\}, \{1, \ell\}\) for each of the within-chain edges and \( \{0, \ell\}, \{0, \ell+1\}\) for each of the between-chain edges. The FCRF \( G \) defines a distribution over hidden states as:

\[
p(y|x) = \frac{1}{Z(x)} \left( \prod_{t=1}^{T-1} \prod_{\ell=1}^{L} \Phi_{\ell}(y_{\ell,t}, y_{\ell,t+1}, x, t) \right) \left( \prod_{t=1}^{T-1} \prod_{\ell=1}^{L-1} \Psi_{\ell}(y_{\ell,t}, y_{\ell+1,t}, x, t) \right),
\]

(4)

where \( \{ \Phi_{\ell} \} \) are the potentials over the within-chain edges, \( \{ \Psi_{\ell} \} \) are the potentials over the between-chain edges, and \( Z(x) \) is the partition function. The potentials factorize according to the features \( \{ f_k \} \) and weights \( \{ \lambda_k \} \) of \( G \) as:

\[
\Phi_{\ell}(y_{\ell,t}, y_{\ell,t+1}, x, t) = \exp \left( \sum_k \lambda_k f_k(y_{\ell,t}, y_{\ell,t+1}, x, t) \right)
\]

\[
\Psi_{\ell}(y_{\ell,t}, y_{\ell+1,t}, x, t) = \exp \left( \sum_k \lambda_k f_k(y_{\ell,t}, y_{\ell+1,t}, x, t) \right)
\]

More complicated structures are also possible, such as semi-Markov CRFs, in which the state transition probabilities depend on how long the chain has been in its current state, and hierarchical CRFs, which are moralized versions of the hierarchical HMMs of Fine et al. (1998).1 As in DBNs, this factorized structure can use many fewer parameters than the cross-product state space: even the two-level FCRF we discuss below uses less than an eighth of the parameters of the corresponding cross-product CRF.

---

1Hierarchical HMMs were shown to be DBNs by Murphy and Paskin (2001).
3.2. Inference in DCRFs

Inference in a DCRF can be done using any inference algorithm for undirected models. For an unlabeled sequence \( x \), we typically wish to solve two inference problems: (a) computing the marginals \( p(y_{t,c} | x) \) over all cliques \( y_{t,c} \), and (b) computing the Viterbi decoding \( y^* = \arg \max_y p(y|x) \). The Viterbi decoding is used to label a new sequence, and marginal computation is used for parameter estimation (Section 3.3).

Because marginal computation is needed during training, inference must be efficient so that we can use large training sets even if there are many labels. The largest experiment reported here required computing pairwise marginals in 866,792 different graphical models: one for each training example in each iteration of a convex optimization algorithm. Since exact inference can be expensive in complex DCRFs, we use approximate methods. Here we describe approximate inference using loopy belief propagation.

Although belief propagation is exact only in certain special cases, in practice it has been a successful approximate method for general graphical models (Murphy et al., 1999; Aji et al., 1998). In general, belief propagation algorithms iteratively update a vector \( \mathbf{m} = (m_u(x_v)) \) of messages between pairs of vertices \( x_u \) and \( x_v \). The update from \( x_u \) to \( x_v \) is given by:

\[
m_u(x_v) \leftarrow \sum_{x_u} \Phi(x_u, x_v) \prod_{x_i \neq x_u} m_t(x_u) \prod_{x_w \neq x_u} m_w(x_v),
\]

where \( \Phi(x_u, x_v) \) is the potential on the edge \((x_u, x_v)\). Performing this update for one edge \((x_u, x_v)\) in one direction is called sending a message from \( x_u \) to \( x_v \). Given a message vector \( \mathbf{m} \), approximate marginals are computed as

\[
p(x_u, x_v) \leftarrow \kappa \Phi(x_u, x_v) \prod_{x_i \neq x_u} m_t(x_u) \prod_{x_w \neq x_u} m_w(x_v),
\]

where \( \kappa \) is a normalization factor.

At each iteration of belief propagation, messages can be sent in any order, and choosing a good schedule can affect how quickly the algorithm converges. We describe two schedules for belief propagation: tree-based and random. The tree-based schedule, also known as tree reparameterization (TRP) (Wainwright et al., 2001; Wainwright, 2002), propagates messages along a set of cross-cutting spanning trees of the original graph. At each iteration of TRP, a spanning tree \( \mathcal{T}^{(i)} \in \Upsilon \) is selected, and messages are sent in both directions along every edge in \( \mathcal{T}^{(i)} \), which amounts to exact inference on \( \mathcal{T}^{(i)} \). In general, trees may be selected from any set \( \Upsilon = \{ \mathcal{T} \} \) as long as the trees in \( \Upsilon \) cover the edge set of the original graph. In practice, we select trees randomly, but we select first edges that have never been used in any previous iteration.

The random schedule simply sends messages across all edges in random order. To improve convergence, we arbitrarily order each edge \( e_i = (s_i, t_i) \) and send all messages \( m_{s_i}(t_i) \) before any messages \( m_{t_i}(s_i) \). Note that for a graph with \( V \) nodes and \( E \) edges, TRP sends \( O(V) \) messages per BP iteration, while the random schedule sends \( O(E) \) messages.

To perform Viterbi decoding, we use the same propagation algorithms, except that the summation in Equation 5 is replaced by maximization. Also, the algorithms that we have described apply to DCRFs with at most pairwise cliques. Inference in DCRFs with larger cliques can be performed straightforwardly using generalized versions of the variational approaches in this section (Yedidia et al., 2000; Wainwright, 2002).

3.3. Parameter Estimation in DCRFs

The parameter estimation problem is to find a set of parameters \( \Lambda = \{ \lambda_k \} \) given training data \( \mathcal{D} = \{ (x^{(i)}, y^{(i)}) \}_{i=1}^{N} \). More specifically, we optimize the conditional log-likelihood

\[
\mathcal{L}(\Lambda) = \sum_{i} \log p_{\Lambda}(y^{(i)} | x^{(i)}).
\]

The derivative of this with respect to a parameter \( \lambda_k \) associated with clique index \( c \) is

\[
\frac{\partial \mathcal{L}}{\partial \lambda_k} = \sum_{i} \sum_{t} f_k(\bar{y}_{t,c}, x^{(i)}; t) - \sum_{i} \sum_{t} \sum_{\bar{y}_{t,c}} p_{\Lambda}(\bar{y}_{t,c} | x^{(i)}) f_k(\bar{y}_{t,c}, x^{(i)}; t),
\]

where \( \bar{y}_{t,c} \) is the assignment to \( y_{t,c} \) in \( y^{(i)} \), and \( \bar{y}_{t,c} \) ranges over assignments to the clique \( y_{t,c} \). Observe that it is the factor \( p_{\Lambda}(\bar{y}_{t,c} | x^{(i)}) \) that requires us to compute marginal probabilities in the unrolled DCRF.

To reduce overfitting, we define a prior \( p(\Lambda) \) over parameters, and optimize \( \log p(\Lambda | \mathcal{D}) = \mathcal{L}(\Lambda) + \log p(\Lambda) \). We use a spherical Gaussian prior with mean \( \mu = 0 \) and covariance matrix \( \Sigma = \sigma^2 I \), so that the gradient becomes

\[
\frac{\partial p(\Lambda | \mathcal{D})}{\partial \lambda_k} = \frac{\partial \mathcal{L}}{\partial \lambda_k} - \frac{\lambda_k}{\sigma^2}.
\]

See Peng and McCallum (2004) for a comparison of different priors for linear-chain CRFs.

The function \( p(\Lambda | \mathcal{D}) \) is convex, and can be optimized by any number of techniques, as in other maximum-entropy models (Lafferty et al., 2001; Berger et al., 1996). In the results below, we use L-BFGS, which has previously outperformed other optimization algorithms for linear-chain CRFs (Sha & Pereira, 2003; Malouf, 2002).

The analysis above was for the fully-observed case, where the training data include observed values for all variables in
the model. If some nodes are unobserved, the optimization problem becomes more difficult, because the log likelihood is no longer convex in general (details omitted for space).

4. Experiments

We present experiments comparing factorial CRFs to other approaches on noun-phrase chunking (Sang & Buchholz, 2000). Also, we compare different schedules of loopy belief propagation in factorial CRFs.

4.1. Noun-Phrase Chunking

Automatically finding the base noun phrases in a sentence can be viewed as a sequence labeling task by labeling each word as either BEGIN-PHRASE, INSIDE-PHRASE, or OTHER (Ramshaw & Marcus, 1995). The task is typically performed by an initial pass of part-of-speech tagging, but then it can be difficult to recover from errors by the tagger. In this section, we address this problem by performing part-of-speech tagging and noun-phrase segmentation jointly in a single factorial CRF.

Our data comes from the CoNLL 2000 shared task (Sang & Buchholz, 2000), and consists of sentences from the Wall Street Journal annotated by the Penn Treebank project (Marcus et al., 1993). We consider each sentence to be a training instance, with single words as tokens. The data are divided into a standard training set of 8936 sentences and a test set of 2012 sentences. There are 45 different POS labels, and the three NP labels.

We compare a factorial CRF to two cascaded approaches, which we call CRF+CRF and Brill+CRF. CRF+CRF uses one linear-chain CRF to predict POS labels, and another linear-chain CRF to predict NP labels, using as a feature the Viterbi POS labeling from the first CRF. Brill+CRF predicts NP labels using the POS labels provided from the Brill tagger, which we expect to be more accurate than those from our CRF, because the Brill tagger was trained on over four times more data, including sentences from the CoNLL 2000 test set.

The factorial CRF uses the graph structure in Figure 1(b), with one chain modeling the part-of-speech process and the other modeling the noun-phrase process. We use L-BFGS to optimize the posterior \( p(\Lambda|D) \), and TRP to compute the marginal probabilities required by \( \partial \mathcal{L}/\partial \lambda_k \). Based on past experience with linear-chain CRFs, we use the prior variance \( \sigma^2 = 10 \) for all models.

We factorize our features as \( f_k(y_{t,c}, x, t) = p_k(y_{t,c}|x, t)q_k(x, t) \) where \( p_k(y_{t,c}|x, t) \) is a binary function on the assignment, and \( q_k(x, t) \) is a function solely of the input string. Table 2 shows the features we use. All three approaches use the same features, with the obvious exception that the FCRF and the first stage of CRF+CRF do not use the POS features \( T_1 = T \).

Performance on noun-phrase chunking is summarized in Table 1. As usual, we measure performance on chunking by \textit{precision}, the percentage of returned phrases that are
Table 2. Input features \( q_k(x, t) \) for the CoNLL data. In the above \( w_t \) is the word at position \( t \), \( T_t \) is the POS tag at position \( t \), \( w \) ranges over all words in the training data, and \( T \) ranges over all part-of-speech tags.

\[
w_t \delta = w \\
w_t \text{ matches } [\text{A-Z}] [a-z] + \\
w_t \text{ matches } [\text{A-Z}] \\
w_t \text{ matches } [\text{A-Z}] + [a-z] + [A-Z] + [a-z] \\
w_t \text{ matches } . *[0-9], * \\
w_t \text{ appears in list of first names, last names, company names, days, months, or geographic entities} \\
w_t \text{ is contained in a lexicon of words with POS } T \text{ (from Brill tagger)} \\
T_t = T \\
q_k(x, t + \delta) \text{ for all } k \text{ and } \delta \in [-3, 3]
\]

\( w_t \delta = w \) ranges over all words in the training data, and \( T \) ranges over all part-of-speech tags.

\[q_k(x, t + \delta) \text{ for all } k \text{ and } \delta \in [-3, 3] \]

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (hr)</th>
<th>NP F1</th>
<th>LBFGS iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random (3)</td>
<td>15.67</td>
<td>2.90</td>
<td>88.37</td>
</tr>
<tr>
<td>Tree (3)</td>
<td>13.85</td>
<td>11.6</td>
<td>88.02</td>
</tr>
<tr>
<td>Tree (\infty)</td>
<td>13.57</td>
<td>3.03</td>
<td>88.67</td>
</tr>
<tr>
<td>Random (\infty)</td>
<td>13.25</td>
<td>1.51</td>
<td>88.60</td>
</tr>
<tr>
<td>Exact</td>
<td>20.49</td>
<td>1.97</td>
<td>88.63</td>
</tr>
</tbody>
</table>

Table 3. Comparison of F1 performance on the chunking task by inference algorithm. The columns labeled \( \mu \) give the mean over five repetitions, and \( s \) the sample standard deviation. Approximate inference methods have labeling accuracy very similar to exact inference with lower total training time. The differences in training time between Tree (\infty) and Exact and between Random (\infty) and Exact are statistically significant by a paired \( t \)-test \( (df = 4; p < 0.005) \).

On this data set, several systems are statistically tied for best performance. Kudo and Matsumoto (2001) report an F1 of 94.39 using a combination of voting support vector machines. Sha and Pereira (2003) give a linear-chain CRF that achieves an F1 of 94.38, using a second-order Markov assumption, and including bigram and trigram POS tags as features. An FCRF imposes a first-order Markov assumption over labels, and represents dependencies only between cotemporal POS and NP label, not POS bigrams or trigrams. Thus, Sha and Pereira’s results suggest that more richly-structured DCRFs could achieve better performance than an FCRF.

Other DCRF structures can be applied to many different language tasks, including information extraction. Peshkin and Pfeffer (2003) apply a generative DBN to extraction from seminar announcements (Fritag & McCallum, 1999), attaining improved results, especially in extracting locations and speakers, by adding a factor to remember the identity of the last non-background label. Our early results with a similar structure seem promising, for example, one DCRF structure performs within 2% F1 of a linear chain CRF, despite being trained on 37% less data.

### 4.2. Comparison of Inference Algorithms

Because DCRFs can have rich graphical structure, and require many marginal computations during training, inference is critical to efficient training with many labels and large data sets. In this section, we compare different inference methods both on training time and labeling accuracy of the final model.

Because exact inference is feasible for a two-chain FCRF, this provides a good case to test whether the final classifica-
tion accuracy suffers when approximate methods are used to calculate the gradient. Also, we can compare different methods for approximate inference with respect to speed and accuracy.

We train factorial CRFs on the noun-phrase chunking task described in the last section. We compute the gradient using exact inference and approximate belief propagation using random, and tree-based schedules, as described in section 3.2. Algorithms are considered to have converged when no message changes by more than $10^{-3}$. In these experiments, the approximate BP algorithms always converged, although this is not guaranteed in general. We trained on five random subsets of 5% of the training data, and the same five subsets were used in each condition. All experiments were performed on a 2.8 GHz Intel Xeon with 4 GB of memory.

For each message-passing schedule, we compare terminating on convergence (Random(∞) and Tree(∞) in Table 3), to terminating after three iterations (Random (3) and Tree (3)). Although the early-terminating BP runs are less accurate, they are faster, which we hypothesized could result in lower overall training time. If the gradient is too inaccurate, however, then the optimization will require many more iterations, resulting in greater training time overall, even though the time per gradient computation is lower. Another hazard is that no maximizing step may be possible along the approximate gradient, even if one is possible along the true gradient. In this case, the gradient descent algorithm terminates prematurely, leading to decreased performance.

Table 3 shows the average F1 score and total training times of DCRFs trained by the different inference methods. Unexpectedly, letting the belief propagation algorithms run to convergence led to lower training time than the early cutoff. For example, even though Random(3) averaged 427 sec per gradient computation compared to 571 sec for Random(∞), Random(∞) took less total time to train, because Random(∞) needed an average of 83.6 gradient computations per training run, compared to 133.2 for Random(3).

As for final classification performance, the various approximate methods and exact inference perform similarly, except that Tree(3) has lower final performance because maximization ended prematurely, averaging only 32.6 maximizer iterations. The variance in F1 over the subsets, although not large, is much larger than the F1 difference between the inference algorithms.

Previous work (Wainwright, 2002) has shown that TRP converges faster than synchronous belief propagation, that is, with Jacobi updates. Both the schedules discussed in section 3.2 use asynchronous Gauss-Seedel updates. We emphasize that the graphical models in these experiments are always pairs of coupled chains. On more complicated models, or with a different choice of spanning trees, tree-based updates could outperform random asynchronous updates. Also, in complex models, the difference in classification accuracy between exact and approximate inference could be larger, but then exact inference is likely to be intractable.

In summary, we draw three conclusions about this model. First, using approximate inference instead of exact inference leads to lower overall training time with no loss in accuracy. Second, there is little difference between a random tree schedule and a completely random schedule for belief propagation. Third, running belief propagation to convergence leads both to increased classification accuracy and lower overall training time than an early cutoff.

5. Conclusions

Dynamic CRFs are conditionally-trained undirected sequence models with repeated graphical structure and tied parameters. They combine the best of both conditional random fields and the widely successful dynamic Bayesian networks (DBNs). DCRFs address difficulties of DBNs, by easily incorporating arbitrary overlapping input features, and of previous conditional models, by allowing more complex dependence between labels. Inference in DCRFs can be done using approximate methods, and training can be done by maximum a posteriori estimation.

Empirically, we have shown that factorial CRFs can be used to jointly perform several labeling tasks at once, sharing information between them. Such a joint model performs better than a model that does the individual labeling tasks sequentially, and has potentially many practical implications, because cascaded models are ubiquitous in NLP. Also, we have shown that using approximate inference leads to lower total training time with no loss in accuracy.

In future research, we plan to explore other inference methods to make training more efficient, including expectation propagation (Minka, 2001) and variational approximations. Also, investigating other DCRF structures, such as hierarchical CRFs and DCRFs with memory of previous labels, could lead to applications into many of the tasks to which DBNs have been applied, including object recognition, speech processing, and bioinformatics.

Acknowledgments

We thank the three anonymous reviewers for many helpful comments. This work was supported in part by the Center for Intelligent Information Retrieval; by SPAWARSYSCEN-SD grant number N66001-02-1-8903; by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract number NBCHD030010; and by the Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant # IIS-0326249. Any opinions, findings and conclusions or recommendations expressed in this material are the au-
thors’ and do not necessarily reflect those of the sponsors.

References


