J. Indian Inst. Sci.
A Multidisciplinary Reviews Journal
ISSN: 0970-4140 Coden-JIISAD

REVIEW ARTICLE

Sequence Segmentation Using Semi-Markov Conditional Random Fields

Sunita Sarawagi[*]

Abstract | Many applications in natural language, speech processing and data integration require model-based segmentation of sequences. Semi-Markov conditional random fields (semi-CRFs) are a generalization of CRFs and provide a full conditional distribution over all possible segmentation of a sequence. Semi-CRFs are particularly suitable for tasks that entail segment-level features such as match with existing dictionary of segments. Empirical results on real-life NER tasks show that they yield higher accuracy than CRFs, but the straightforward foreword–backward inference algorithm requires 3–10 times the computation cost of CRFs. This running time can be reduced significantly by exploiting overlapping features across segments. We present a succinct representation of overlapping features and an efficient training algorithm that can sum over all possible input segmentation in time that is sub-quadratic in the input length, even while imposing no bound on the maximum segment length. Consequently, the running time becomes comparable to CRFs even with the addition of useful entity-level features on large input segments.

1 Introduction

Sequence segmentation models form the core of several applications including speech segmentation on phoneme boundaries[11], information extraction[20], named entity recognition[18], syntactic chunking, shallow parsing, pitch accent prediction, and, protein/gene finding. Traditionally, many of these applications have been artificially formulated as sequence labeling tasks at the expense of loss of flexibility of features that can be exploited. This limitation is partly addressed by expanding the label set—for example, a popular choice in named entity recognition tasks (NER) is the Begin-Continue-End-Unique-other (BCEUO) encoding of entity labels[3], and in syntactic chunking tasks is the Begin-Inside-Outside (BIO) encoding of labels[24]. However, with the increasing diversity of settings where many of the applications are being deployed, it is imperative to exploit a richer variety of features than has been possible by sequence models. Examples of such segment-level features for extraction tasks are: the whole entity has an exact match in a database of entities, the length of the entity is between four and eight words, and the third or fourth token of the entity is a "-". Sequence segmentation models provide a direct and natural way of encapsulating all such entity features. A second advantage is that a joint distribution over segmentation makes it easy to cascade a segmentation model with other tasks, such as deduplication of the extracted entities. This enables us to express data integration as an end-to-end task that combines the steps of information extraction and deduplication. Finally, the resultant data can be represented as a probabilistic database under the row/column uncertainty model.

Formal definition of sequence segmentation Given an input sequence $\boldsymbol{x} = x_1 \ldots x_n$, a segmentation $\boldsymbol{s}$ of $\boldsymbol{x}$ consists of a sequence of variable length segments $\boldsymbol{s} = \langle s_1, \ldots, s_p \rangle$ where each segment $s_j = \langle t_j, u_j, y_j \rangle$ consists of a *start position* t_j, an *end position* u_j, and a *label* $y_j \in Y$. Conceptually, a segment means that the tag y_j is given to all x_i's between $i = t_j$ and $i = u_j$, inclusive. Each segment s_j can be associated with a vector of features

IIT Bombay, Mumbai, India.
**sunita@iitb.ac.in*

Springer IISc Press

that captures the dependence of its label on input properties in the neighborhood of the segment and the label of the segment before it. The goal during inference is to simultaneously find a segmentation of the input sequence and label each segment so as to maximize the total score over all segments.

We propose to model the problem of sequence segmentation using semi-Markov conditional random fields (or semi-CRFs for short), a conditionally trained version of semi-Markov chains. In Sect. 2, we define the semi-CRF model and the corresponding training and inference algorithms. A limitation of the semi-CRF model over CRFs is the increased inference complexity. In Sect. 3, we present an inference algorithm that exploits the common case of a segment score being aggregated over much smaller sub-segment scores that are shared across segmentation. In Sect. 4, we present diverse applications of segmentation models.

2 The Semi-CRF Model

We start this section by first giving an overview of conditional random fields (CRFs) based on which semi-CRFs are designed.

CRFs A CRF[13] models $\Pr(\boldsymbol{y}|\boldsymbol{x})$ using a Markov random field, with nodes corresponding to elements of the structured object $\boldsymbol{y}$, and potential functions that are conditional on (features of) $\boldsymbol{x}$. Learning is performed by setting parameters to maximize the likelihood of a set of $(\boldsymbol{x}, \boldsymbol{y})$ pairs given as training data. One common use of CRFs is for sequential learning problems like NP chunking[22], POS tagging[13], and NER[17]. For these problems, the Markov field is a chain, and $\boldsymbol{y}$ is a linear sequence of labels from a fixed set $\mathcal{Y}$. For instance, in the NER application, $\boldsymbol{x}$ might be a sequence of words, and $\boldsymbol{y}$ might be a sequence in $\{I, O\}^{|\boldsymbol{x}|}$, where $y_i = I$ indicates "word x_i is inside a name" and $y_i = O$ indicates the opposite. Assume a vector $\boldsymbol{f}$ of *local feature functions* $\boldsymbol{f} = \langle f^1, \ldots, f^K \rangle$, each of which maps a pair $(\boldsymbol{x}, \boldsymbol{y})$ and an index i to a measurement $f^k(i, \boldsymbol{x}, \boldsymbol{y}) \in R$. Let $\boldsymbol{f}(i, \boldsymbol{x}, \boldsymbol{y})$ be the vector of these measurements, and let $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}) = \sum_i^{|\boldsymbol{x}|} \boldsymbol{f}(i, \boldsymbol{x}, \boldsymbol{y})$. For the case of NER, the components of $\boldsymbol{f}$ might include the measurement $f^{13}(i, \boldsymbol{x}, \boldsymbol{y}) = [\![x_i \text{ is capitalized}]\!] \cdot [\![y_i = I]\!]$, where the indicator function $[\![c]\!] = 1$ if c if true and zero otherwise; this implies that $F^{13}(\boldsymbol{x}, \boldsymbol{y})$ would be the number of capitalized words x_i paired with the label I. Following previous work[13, 22], we will define a conditional random field (CRF) to be an estimator of the form:

$$\Pr(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W}) = \frac{1}{Z(\boldsymbol{x})} e^{\boldsymbol{W} \cdot \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})}, \tag{1}$$

where $\boldsymbol{W}$ is a weight vector over the components of $\boldsymbol{F}$, and $Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}'} e^{\boldsymbol{W} \cdot \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}')}$.

To extend this to the semi-Markov case, let $\boldsymbol{s} = \langle s_1, \ldots, s_p \rangle$ denote a *segmentation* of $\boldsymbol{x}$, where *segment* $s_j = \langle t_j, u_j, y_j \rangle$ consists of a *start position* t_j, an *end position* u_j, and a *label* $y_j \in Y$. Conceptually, a segment means that the tag y_j is given to all x_i's between $i = t_j$ and $i = u_j$, inclusive. We assume segments have positive length, and adjacent segments touch, that is t_j and u_j always satisfy $1 \leq t_j \leq u_j \leq |\boldsymbol{x}|$, $t_{j+1} = u_j + 1$, $u_p = |\boldsymbol{x}|$, and $t_1 = 1$.

We assume a vector $\boldsymbol{g}$ of *segment feature functions* $\boldsymbol{g} = \langle g^1, \ldots, g^K \rangle$, each of which maps a triple $(j, \boldsymbol{x}, \boldsymbol{s})$ to a measurement $g^k(j, \boldsymbol{x}, \boldsymbol{s}) \in R$, and define $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{s}) = \sum_j^{|\boldsymbol{s}|} \boldsymbol{g}(j, \boldsymbol{x}, \boldsymbol{s})$. We also make a restriction on the features, analogous to the usual Markovian assumption made in CRFs, and assume that every component g^k of $\boldsymbol{g}$ is a function only of $\boldsymbol{x}$, s_j, and the label y_{j-1} associated with the preceding segment s_{j-1}. In other words, we assume that every $g^k(j, \boldsymbol{x}, \boldsymbol{s})$ can be rewritten as:

$$g^k(j, \boldsymbol{x}, \boldsymbol{s}) = g'^k(y_j, y_{j-1}, \boldsymbol{x}, t_j, u_j), \tag{2}$$

for an appropriately defined g'^k. In the rest of the paper, we will drop the g' notation and use g for both versions of the segment-level feature functions. A *semi-CRF* is then an estimator of the form:

$$\Pr(\boldsymbol{s}|\boldsymbol{x}, \boldsymbol{W}) = \frac{1}{Z(\boldsymbol{x})} e^{\boldsymbol{W} \cdot \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{s})}, \tag{3}$$

where again $\boldsymbol{W}$ is a weight vector for $\boldsymbol{G}$ and $Z(\boldsymbol{x}) = \sum_{\boldsymbol{s}'} e^{\boldsymbol{W} \cdot \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{s}')}$. Thus, we get a probability distribution over all possible labeled segmentation of an input sequence.

2.1 Prediction

Given $\boldsymbol{W}$ and $\boldsymbol{x}$, we predict the segmentation with the highest probability, that is, $\mathit{argmax}_{\boldsymbol{s}} \Pr(\boldsymbol{s}|\boldsymbol{x}, \boldsymbol{W})$, where $\Pr(\boldsymbol{s}|\boldsymbol{x}, \boldsymbol{W})$ is defined by Eq. 3. An efficient inference algorithm is suggested by Eq. 2, which implies that:

$$\begin{aligned} &\operatorname{argmax}_{\boldsymbol{s}} \Pr(\boldsymbol{s}|\boldsymbol{x}, \boldsymbol{W}) \\ &\quad = \operatorname{argmax}_{\boldsymbol{s}} \boldsymbol{W} \cdot \sum_j \boldsymbol{g}(y_j, y_{j-1}, \boldsymbol{x}, t_j, u_j), \end{aligned}$$

Let L be an upper bound on segment length. Let $\boldsymbol{s}_{i:y}$ denote the set of all partial segmentation starting from 1 (the first index of the sequence) to i, such that the last segment has the label y and ending position i. Let $V_{\boldsymbol{x}, \boldsymbol{g}, \boldsymbol{W}}(i, y)$ denote

the largest value of $\boldsymbol{W} \cdot \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{s}')$ for any $\boldsymbol{s}' \in \boldsymbol{s}_{i:y}$. Omitting the subscripts, the following recursive calculation implements a semi-Markov analog of the usual Viterbi algorithm:

$$V(i,y) = \begin{cases} \max\limits_{y', i-L<j<i} V(i-d, y') \\ \quad + \boldsymbol{W} \cdot \boldsymbol{g}(y, y', \boldsymbol{x}, j, i) & \text{if } i > 0 \\ 0 & \text{if } i = 0 \\ -\infty & \text{if } i < 0 \end{cases},$$

The best segmentation then corresponds to the path traced by $\max_y V(|\boldsymbol{x}|, y)$.

2.2 Training

During training, the goal is to maximize log likelihood over a given training set $T = \{(\boldsymbol{x}_\ell, \boldsymbol{s}_\ell)\}_{\ell=1}^{N}$. Following the notation of Sha and Pereira[22], we express the log likelihood over the training sequences as:

$$L(\boldsymbol{W}) = \sum_\ell (\boldsymbol{W} \cdot \boldsymbol{G}(\boldsymbol{x}_\ell, \boldsymbol{s}_\ell) - \log Z_{\boldsymbol{W}}(\boldsymbol{x}_\ell)), \quad (4)$$

We wish to find a $\boldsymbol{W}$ that maximizes $L(\boldsymbol{W})$. Equation 4 is convex, and can thus be maximized by gradient ascent, or one of many related methods. (In our implementation we use a limited-memory quasi-Newton method[14, 15].) The gradient of $L(\boldsymbol{W})$ is the following:

$$\begin{aligned} \nabla L(\boldsymbol{W}) &= \sum_\ell \boldsymbol{G}(\boldsymbol{x}_\ell, \boldsymbol{s}_\ell) - \frac{\sum_{\boldsymbol{s}'} \boldsymbol{G}(\boldsymbol{s}', \boldsymbol{x}_\ell) e^{\boldsymbol{W} \cdot \boldsymbol{G}(\boldsymbol{x}_\ell, \boldsymbol{s}')}}{Z_{\boldsymbol{W}}(\boldsymbol{x}_\ell)} \\ &= \sum_\ell \boldsymbol{G}(\boldsymbol{x}_\ell, \boldsymbol{s}_\ell) - E_{\Pr(\boldsymbol{s}'|\boldsymbol{W})} \boldsymbol{G}(\boldsymbol{x}_\ell, \boldsymbol{s}') \end{aligned},$$

The two computationally expensive terms above are the normalizer, $Z_{\boldsymbol{W}}(\boldsymbol{x}_\ell)$, and the expected value of the features under the current weight vector. We describe an algorithm much like the sum-product message passing algorithm in graphical models for computing these terms next.

Computing $Z_{\boldsymbol{W}}(\boldsymbol{x}_\ell)$ and $E_{\Pr(\boldsymbol{s}'|\boldsymbol{W})} \boldsymbol{G}(\boldsymbol{x}_\ell, \boldsymbol{s}')$. We assume the feature vector $\boldsymbol{g}(y, y', \boldsymbol{x}, j, i)$ can be partitioned as 'edge' features that do not depend on i, and segment-level features that do not depend on previous label y'. Using these two types of features, we adopt the potential notation that is common in graphical models, and define two types of potentials:

- Segment potentials $\theta_{i:j}(y)$ associated with a segment from i to j where all nodes from i to j have the base label y. Such potentials can be expressed in terms of segment features as:

$$\theta_{i:j}(y) = \exp\left(\sum_{k \in \text{segment-feature}} w_k \boldsymbol{g}_k(y, \boldsymbol{x}, i, j) \right),$$

  where w_k denotes the weight parameter of feature $\boldsymbol{g}_k$.
- Transition potentials $\boldsymbol{\theta}_i$ where an entry $\boldsymbol{\theta}_i(y', y)$ denotes the potential for a segment starting at i getting a label y when the previous segment is labeled y'. Such potentials can be expressed in terms of edge features as:

$$\boldsymbol{\theta}_i(y', y) = \exp\left(\sum_{k \in \text{edge feature}} w_k \boldsymbol{g}_k(y, y', \boldsymbol{x}, i,) \right).$$

With these two types of potentials, the total score of a segmentation $\boldsymbol{s} = \langle s_1, \ldots, s_p \rangle$ where each segment $s_j = \langle t_j, u_j, y_j \rangle$ consists of a start position t_j, an end position u_j, and a label $y_j \in Y$ can be expressed as $\prod_{j=1}^{p} \boldsymbol{\theta}_{t_j}(y_{j-1}, y_j) \theta_{t_j:u_j}(y_j)$. The marginal probability of any segment potential can be computed via forward $\boldsymbol{\alpha}$ and backward $\boldsymbol{\beta}$ messages which are expressed recursively as follows:

Let $\alpha_i(y)$ denote the sum of scores over all segmentation of the sequence between 1 to i where the last segment has a label y.

$$\alpha_i(y) = \sum_{\max(i-L,1) \le i' \le i} \sum_{y' \in \mathcal{Y}} \alpha_{i'-1}(y') \theta_{i'}(y', y) \theta_{i':i}(y). \quad (5)$$

The running time of this algorithm is $O(nL^2)$ where n is the input sequence length. This is because there are $O(nL)$ iterations and each segment $i' : i$ requires $O(i - i')$ work in computing the potential $\boldsymbol{\theta}_{i':i}$.

Similarly, we can recursively compute the backward messages $\beta_i(y)$ that denotes the sum of scores over all segmentation of the sequence from $i+1$ to n with the segment ending at i having label y. Using the α and β values we can compute the normalizer $Z(\boldsymbol{x}) = \sum_y \alpha_n(y)$ and the marginal $\mu_{i':i}(y)$ for a segment potential $i' : i$ labeled y as:

$$\mu_{i':i}(y) = \frac{\sum_{y'} \alpha_{i'-1}(y') \theta_{i'}(y', y) \theta_{i':i}(y) \beta_i(y)}{Z(\boldsymbol{x})}.$$

The expected value of a segment feature $\boldsymbol{g}_k$ is:

$$E_{\Pr(\boldsymbol{s}'|\boldsymbol{W})} \boldsymbol{g}_k(\boldsymbol{x}_\ell, \boldsymbol{s}') = \sum_{(i',i,y)} \mu_{i':i}(y) \boldsymbol{g}_k(y, , \boldsymbol{x}, i', i).$$

Likewise, for edge features. Thus, in $O(nL^2)$ time and two sets of messages of $O(n)$ length we can compute the marginal probability of any potential in the segmentation model.

Empirically, for typical NER tasks, segmentation models are found to require about 3–10 times the running time of sequential models. The value of the maximum length parameter L has to be chosen to be large enough to cover the largest segment because it is a hard limit on the

length of the segment. Thus, for extraction from short text snippets, for example, citation records, L becomes comparable to n to cover long entities like "Titles". This makes inference in segmentation models cubic in the length of the sequence. We present a more efficient algorithm for solving this inference problem in Sect. 3.

The semi-CRF has been shown to provide better accuracy than CRFs for several information extraction tasks. More details can be found in Sarawagi and Cohen[20].

3 Overlapping forward–backward algorithm

We next show how to improve inference speed of segmentation models. We call our algorithm "Overlapping forward–backward" algorithm for reasons that will soon become clear. A key insight we employ is that potentials for segmentation should be defined over a larger equivalence class of segments rather than for a single segment. We design a representation to express these equivalences succinctly through four kinds of potentials.

- $\psi_{i':\geq i}$ denotes potentials shared over all segments starting at i' but ending anywhere after or at i.
- $\psi_{\leq i':i}$ denotes potentials shared over all segments ending at i but starting anywhere before or at i'.
- $\psi_{\leq i':\geq i}$ denotes potentials shared over all segments starting before or at i' and ending after or at i.
- $\psi_{i':i}$ denotes the usual full segment potentials that apply to the exact segment between i' and i.

We list some examples of such potentials in the context of a NER task:

- First two tokens of segment starting at i are "Prof. Dr": $\psi_{i:\geq i+2}$.
- Last three tokens of segment ending at i are "of the ACM": $\psi_{\leq i-3:i}$.
- A segment of length l starting at i has high cosine similarity to a lexicon entry: $\psi_{i:i+l-1}$.
- Segment contains "the" and "the" is the i-th word in $\boldsymbol{x}$: $\psi_{\leq i:\geq i}$.
- The length of a segment starting at i is $> k$: $\psi_{i:\geq i+k}$.

Our goal next is to exploit the succinct form of potentials to speed up the message passing algorithms outlined in Sect. 2.1. Our new message passing algorithms can run in time that in the worst case is $O(nm + H)$ where m is the largest gap between the start and end boundary of potentials in *that sequence* and is typically smaller than the largest length of a segment and, H is the total number of potentials that are fired for a sequence. Thus, for sequence labeling tasks where $m = 2$, this will reduce to the standard $O(n)$ forward–backward algorithm, even though it can potentially output segments much larger than 2. We generalize this to the case of an arbitrary set of potentials. The main challenge in designing an efficient algorithm is that potentials could overlap in arbitrary ways and we cannot afford to pass messages only along transition edges as we did in Eq. 5 with only full segment potentials.

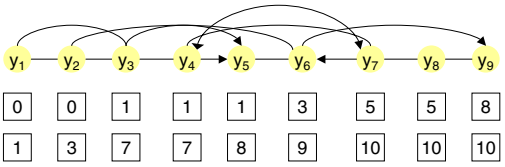

Figure 1: Potentials for a sequence of length 9. Potentials of the form $\psi_{j:\geq i}$ are represented with edges having an arrow at i, potentials of the form $\psi_{\leq j:i}$ are represented with edges having an arrow at j and so on. The first row of integers denotes $A(i)$ values for each i and the second row denotes the $B(i)$ values for each i. In this case $m = 4$.

We first show how this is done for forward and backward messages in Sect. 3.1. We then show in Sect. 3.2 how to directly compute marginals over potentials instead of summing over marginal probability of all segments the potential overlaps with.

3.1 Forward and Backward Terms

Let $\theta_{i':i}$ denote the product of all potentials applicable to segment $i' : i$; this can be expressed as $\theta_{i':i} = \prod\{\psi_{uv} : u = i' \vee u = (\leq k), k \geq i', v = i \vee v = (\geq j), j \leq i\}$. For example, in Fig. 1 we show various potentials (as edges) for a sequence of length 9 where an arrow at any of the ends of an edge denotes potentials with open ends. Thus, the edge between nodes 3 and 5 denotes potential $\psi_{3:\geq 5}$, whereas the two edges between 4 and 7 denote the potential $\psi_{\leq 4:\geq 7}$ for the arrowed edge and $\psi_{4:7}$ for the plain edge. For segment 4:7 the total potential $\theta_{4:7} = \psi_{4:\geq 5}\psi_{4:7}\psi_{\leq 4:\geq 7}\psi_{\leq 6:7}$

Equation 5 for computing α values with no L restriction can be written in matrix notation as:

$$\boldsymbol{\alpha}_i = \sum_{i' \leq i} (\boldsymbol{\alpha}_{i'-1}\boldsymbol{\theta}_{i'}) * \boldsymbol{\theta}_{i':i} \tag{6}$$

where the symbol "⋆" denotes element-wise multiplication of vectors of the same size. In the rest of the paper, we will drop the "⋆" to reduce clutter and assume it to be implicitly present when two vectors of the same length abut.

Our goal is to reuse computations performed for α_{i-1} to reduce the number of terms summed over for computing α_i. For this, we design a method for decomposing the full segment potentials $\boldsymbol{\theta}_{i':i}$ as $a'(i', i-1)a(i)$ where $a(i)$ is independent of i' and a' only involves potentials with the end boundary at $i-1$. We cannot hope to achieve this in general for all $i' \leq i-1$. Therefore, we define a function $A(i)$ such that for the reduced set $i' \leq A(i-1)$ such a decomposition exists. We show how to design these functions.

Let $A(i)$ be the maximum index $j < i$ such that all potentials that start before or at j end before $i+1$.

$$A(i) = \max j : \forall \boldsymbol{\psi}_{\ell:k} \;\; \ell \leq j \Rightarrow k \leq i.$$

For the example in Fig. 1 $A(6) = 3, A(7) = A(8) = 5$ and so on.

Let $\boldsymbol{\theta}_{i':>i}$ denote the product of potentials common to all segments where the start boundary is i' and end boundary anywhere after i. Thus: $\boldsymbol{\theta}_{i':>i-1} = \prod\{\boldsymbol{\psi}_{r:s} : r = i' \vee r = (\leq j), j \geq i', s = (\geq k), k \leq i-1\}$.

Let $\boldsymbol{\theta}_{i':(>i-1\to i)}$ denote the product of potentials common to all segments that start at i' and end at i but not before i. That is, $\boldsymbol{\theta}_{i':(>i-1\to i)} = \prod\{\boldsymbol{\psi}_{r:s} : r = i' \vee r = (\leq j), j \geq i', s = i \vee s = (\geq i)\}$. Note,

$$\boldsymbol{\theta}_{i':i} = \boldsymbol{\theta}_{i':>i-1}\boldsymbol{\theta}_{i':(>i-1\to i)}. \tag{7}$$

From the definition of $A(i-1)$ we can claim that,

$$\boldsymbol{\theta}_{i':(>i-1\to i)} = \boldsymbol{\theta}_{A(i-1):(>i-1\to i)} \quad \text{if } i' \leq A(i-1). \tag{8}$$

Using Eqs. 7 and 8, we can choose $a(i) = \boldsymbol{\theta}_{A(i-1):(>i-1\to i)}$ and $a'(i', i-1) = \boldsymbol{\theta}_{i':>i-1}$ and use these to write the expression for α_i that reuses computations from α_{i-1} as follows: Let $\hat{\boldsymbol{\alpha}}_i = \boldsymbol{\alpha}_{i-1}\boldsymbol{\theta}_i$ and $\hat{\boldsymbol{\alpha}}_1 = \mathbf{1}$. Equation 6 can be rewritten as:

$$\begin{aligned}\boldsymbol{\alpha}_i &= \sum_{i' \leq A(i-1)} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{i':>i-1}\boldsymbol{a}_i + \sum_{A(i-1)<i' \leq i} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{i':i} \\ &= \boldsymbol{\alpha}^P_{i-1}\boldsymbol{a}_i + \sum_{A(i-1)<i' \leq i} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{i':i}\end{aligned}, \tag{9}$$

where $\boldsymbol{\alpha}^P_{i-1}$ denotes the sum of scores over all possible segmentation of sequence from 1 to i where the last segment's start boundary is before or at $A(i-1)$ and the end boundary after $i-1$. We compute $\boldsymbol{\alpha}^P_i$ recursively as:

$$\boldsymbol{\alpha}^P_i = \begin{cases} \boldsymbol{\alpha}^P_{i-1}\boldsymbol{a}_{>i} + \sum_{j=A(i-1)+1}^{A(i)} \hat{\boldsymbol{\alpha}}_j\boldsymbol{\theta}_{j:>i} & \text{if } A(i) > 0 \\ 0 & \text{if } A(i) \leq 0 \end{cases},$$

where $\boldsymbol{a}_{>i}$ is like $\boldsymbol{a}_i$ except that we exclude potentials where the end boundary is strictly at i.

Thus, by maintaining an additional set of n forward terms denoting $\boldsymbol{\alpha}^P_i$ we are able to compute α_i by summing over only $i - A(i-1)$ terms instead of $i-1$ terms. Note that $i - A(i-1) \leq m$, the maximum gap between the boundaries of any potential.

Example For the potentials in Fig. 1 we show how to compute $\boldsymbol{\alpha}_7$ given $A(6) = 3$.

$$\begin{aligned}\boldsymbol{\alpha}_7 &= \boldsymbol{\alpha}^P_6\boldsymbol{a}_7 + \hat{\boldsymbol{\alpha}}_4\boldsymbol{\theta}_{4:7} + \hat{\boldsymbol{\alpha}}_5\boldsymbol{\theta}_{5:7} + \hat{\boldsymbol{\alpha}}_6\boldsymbol{\theta}_{6:7} \quad \text{(see Eq. 9)} \\ &= \boldsymbol{\alpha}^P_6\boldsymbol{\psi}_{\leq 4:\geq 7}\boldsymbol{\psi}_{\leq 6:7} + \hat{\boldsymbol{\alpha}}_4\boldsymbol{\psi}_{4:\geq 5}\boldsymbol{\psi}_{4:7}\boldsymbol{\psi}_{\leq 4:\geq 7}\boldsymbol{\psi}_{\leq 6:7} \cdot \\ &\quad + \hat{\boldsymbol{\alpha}}_5\boldsymbol{\psi}_{\leq 6:7} + \hat{\boldsymbol{\alpha}}_6\boldsymbol{\psi}_{\leq 6:7}\end{aligned}$$

Similarly, with $A(7) = 5$ we compute $\boldsymbol{\alpha}^P_7$ as

$$\boldsymbol{\alpha}^P_7 = \boldsymbol{\alpha}^P_6\boldsymbol{\psi}_{\leq 4:\geq 7} + \hat{\boldsymbol{\alpha}}_4\boldsymbol{\psi}_{4:\geq 5}\boldsymbol{\psi}_{\leq 4:\geq 7} + \hat{\boldsymbol{\alpha}}_5.$$

Beta terms The backward message $\boldsymbol{\beta}$, for segmentation models is defined as:

$$\boldsymbol{\beta}_i = \boldsymbol{\theta}_{i+1}\sum_{i'>i} \boldsymbol{\theta}_{i+1:i'}\boldsymbol{\beta}_{i'}.$$

The computation of the beta messages can be optimized similarly to reuse terms from $\boldsymbol{\beta}_{i+1}$ in the computation of $\boldsymbol{\beta}_i$. Accordingly, we define an index $B(i)$ such that for all $i' \geq B(i)$ we can decompose $\boldsymbol{\theta}_{i:i'}$ as $b(i)b'(i+1, i')$ where $b(i)$ is independent of i' and b' involves potentials not before $i+1$.

Let $B(i)$ be the smallest index j such that all potentials that end after or at j do not have their start boundaries before i, that is,

$$B(i) = \min j : \forall \boldsymbol{\psi}_{\ell:k} \quad k \geq j \Rightarrow \ell \geq i.$$

The last row in Fig. 1 shows $B(.)$ values. For example $B(4) = 7$ because there is no potential after position 7 that starts before position 4.

Let $\boldsymbol{\theta}_{<i+1:i'}$ denote the product of all potentials in segments that end at i' but start anywhere to the left of $i+1$. That is, $\boldsymbol{\theta}_{<i+1:i'} = \prod\{\boldsymbol{\psi}_{r:s} : r = (\leq k), k \geq i+1, s = i' \vee s = (\geq j), j \leq i'\}$

Let $\boldsymbol{\theta}_{<i+1\to i:i'}$ denote the product of all potentials shared by segments ending at i' and starting at i but not after i. Thus, $\boldsymbol{\theta}_{<i+1\to i:i'} = \prod\{\boldsymbol{\psi}_{r:s} : r = i \vee r = (\leq i), s = i' \vee s = i' \vee s = (\geq j), j \leq i'\}$. Thus,

$$\boldsymbol{\theta}_{i:i'} = \boldsymbol{\theta}_{<i+1\to i:i'}\boldsymbol{\theta}_{<i+1:i'}. \tag{10}$$

The definition of $B(i)$ implies that:

$$\boldsymbol{\theta}_{<i+1\rightarrow i:i'} = \boldsymbol{\theta}_{<i+1\rightarrow i:B(i+1)} \quad \text{if } i' \geq B(i+1). \tag{11}$$

Using Eqs. 10 and 11 we can set $b(i) = \boldsymbol{\theta}_{<i+1\rightarrow i:B(i+1)}$ and $b'(i+1, i') = \boldsymbol{\theta}_{<i+1:i'}$ to compute β_i without summing over all $n-i$ as follows:

$$\begin{aligned}\boldsymbol{\beta}_i &= \boldsymbol{\theta}_{i+1}\left(\sum_{B(i+2)>i'>i} \boldsymbol{\theta}_{i+1:i'}\boldsymbol{\beta}_{i'} + \sum_{i'\geq B(i+2)} \boldsymbol{b}_{i+1}\boldsymbol{\theta}_{<i+2:i'}\boldsymbol{\beta}_{i'}\right) \\ &= \boldsymbol{\theta}_{i+1}\left(\sum_{B(i+2)>i'>i} \boldsymbol{\theta}_{i+1:i'}\boldsymbol{\beta}_{i'} + \boldsymbol{b}_{i+1}\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{i+2}}\right)\end{aligned} \tag{12}$$

where $\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{i}} = \sum_{i'\geq B(i)} \boldsymbol{\theta}_{<i:i'}\boldsymbol{\beta}_{i'}$, represents the sum over all segmentation $\boldsymbol{s}$ where the first segment in $\boldsymbol{s}$ starts before i and ends at or after position $B(i)$. $\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{i}}$ is computed recursively as:

$$\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{i}} = \begin{cases} \boldsymbol{b}_{<i}\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{i+1}} + \sum_{j=B(i)}^{j=B(i+1)-1} \boldsymbol{\theta}_{<i:j}\boldsymbol{\beta}_j & \text{if } B(i) \leq n \\ 0 & \text{otherwise.} \end{cases}$$

where $\boldsymbol{b}_{<i}$ is like $\boldsymbol{b}_i$ except that we exclude potentials where the start boundary is strictly at i.

Most likely segmentation The computation of the most likely segmentation can also optimized via two dynamic programming equations similar to the two equations for $\boldsymbol{\alpha}_i$ and $\boldsymbol{\alpha}^P_i$ above so as to compute the maximum over at most m terms.

3.2 Computing Marginals Around Potentials

The forward and backward messages can be combined to compute the marginals for various potentials. For normal segmentation models with potentials only of the form $\boldsymbol{\psi}_{s:e}$, this involves only $O(1)$ computations of the form $\hat{\boldsymbol{\alpha}}_s\boldsymbol{\theta}_{s:e}\boldsymbol{\beta}_e$. However, for potentials where the segment start and end boundaries are not fixed, computing the marginal for a potential of the form $\boldsymbol{\psi}_{\leq s:\geq e}$ could require summing $O(n^2)$ such terms. We propose two ways to reduce the number of terms to be summed over. First, we use tricks like in the computation of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ terms where we decompose potentials and depend on a parallel set of $\boldsymbol{\alpha}^P$ and $\boldsymbol{\beta}^{\boldsymbol{P}}$ terms. Second, we share computations across adjacent potentials. These two techniques together allow us to compute each marginal in $O(1)$ amortized time per potential. In the following sections we will use $\boldsymbol{\mu}$ to denote un-normalized marginals, i.e., the marginals before the division by $Z(\boldsymbol{x})$.

Potentials of the form $\boldsymbol{\psi}_{\leq s:e}$ For such potentials the marginal $\boldsymbol{\mu}_{\leq s:e}$ requires summing over s terms as follows:

$$\boldsymbol{\mu}_{\leq s:e} = \sum_{i'\leq s} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{i':e}\boldsymbol{\beta}_e.$$

We simplify this computation to reuse work across multiple related marginals as follows:

$$\boldsymbol{\mu}_{\leq s:e} = \begin{cases} \boldsymbol{\alpha}^P_{e-1}\boldsymbol{a}_e\boldsymbol{\beta}_e & \text{if } A(e-1) = s \\ \boldsymbol{\mu}_{\leq(s-1):e} + \hat{\boldsymbol{\alpha}}_s\boldsymbol{\theta}_{s:e}\boldsymbol{\beta}_e & \text{if } A(e-1) < s \end{cases}.$$

The first case in the above equation follows from Eq. 9 used to simplify the computation of $\boldsymbol{\alpha}$ terms. The second case is a simple recursion and shows how we can reuse work in computing marginals of adjacent potentials. By the definition of $A(.)$ in Eq. 8, we know that $s > A(e-1)$ and the case of $A(e-1) = s$ is a base case of the recursion.

Potentials of the form $\boldsymbol{\psi}_{s:\geq e}$. In this case marginals require summing over $n-e$ terms as follows:

$$\boldsymbol{\mu}_{s:\geq e} = \hat{\boldsymbol{\alpha}}_s \sum_{i\geq e} \boldsymbol{\theta}_{s:i}\boldsymbol{\beta}_i.$$

We simplify this computation so as to require summing over no more than m terms as follows:

$$\boldsymbol{\mu}_{s:\geq e} = \begin{cases} \hat{\boldsymbol{\alpha}}_s\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s+1}}\boldsymbol{b}_s & \text{if } B(s+1) = e \\ \boldsymbol{\mu}_{s:\geq(e+1)} + \hat{\boldsymbol{\alpha}}_s\boldsymbol{\theta}_{s:e}\boldsymbol{\beta}_e & \text{if } B(s+1) > e \end{cases}.$$

Edge potentials also fall in this class except that we need to restrict the previous label.

Potentials of the form $\boldsymbol{\psi}_{\leq s:\geq e}$ In this case we need

$$\boldsymbol{\mu}_{\leq s:\geq e} = \sum_{i'\leq s} \hat{\boldsymbol{\alpha}}_{i'} \sum_{i\geq e} \boldsymbol{\theta}_{i':i}\boldsymbol{\beta}_i.$$

We simplify this computation so as to not require summing over $O(n^2)$ terms as follows:

$$\boldsymbol{\mu}_{\leq s:\geq e} = \begin{cases} \boldsymbol{\mu}_{\leq s:\geq(e+1)} + \boldsymbol{\mu}_{\leq s:e} & \text{if } B(s+1) > e \\ (\boldsymbol{\mu}'(s-1)\boldsymbol{b}_{<s} + \hat{\boldsymbol{\alpha}}_s\boldsymbol{b}_s)\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s+1}} & \text{if } B(s+1) = e \end{cases},$$

where $\boldsymbol{\mu}'(s-1) = \boldsymbol{\mu}_{\leq s-1,\geq B(s)}/\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s}}$ In the equation above the first case is obvious. The second case where $B(s+1) = e$ is derived next.

$$\begin{aligned}\boldsymbol{\mu}_{\leq s:\geq e} &= \sum_{i'\leq s} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{<s+1\rightarrow i':e} \sum_{i\geq e} \boldsymbol{\theta}_{<s+1:i}\boldsymbol{\beta}_i \quad \text{(see Eq : 12)} \\ &= \sum_{i'\leq s} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{<s+1\rightarrow i':e}\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s+1}} \\ &= \sum_{i'\leq s} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{<i'+1\rightarrow i':e} \prod_{i'<j\leq s} \boldsymbol{\theta}_{<j+1\rightarrow<j:e}\,\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s+1}} \\ &= \sum_{i'\leq s} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{b}_{i'} \prod_{i'<j\leq s} \boldsymbol{b}_{<j}\,\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s+1}} \quad \text{(see Eq. 11)} \\ &= ((\boldsymbol{\mu}_{\leq s-1:\geq B(s)}/\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s}})\boldsymbol{b}_{<s} + \hat{\boldsymbol{\alpha}}_s\boldsymbol{b}_s)\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s+1}}\end{aligned}$$

It was tricky to get all the indices right in the above algorithm. We verified correctness by testing[1] that we get the same results on a direct com-

[1] Be careful about using the following code—I've only proven that it works, I haven't tested it. *Donald Knuth*

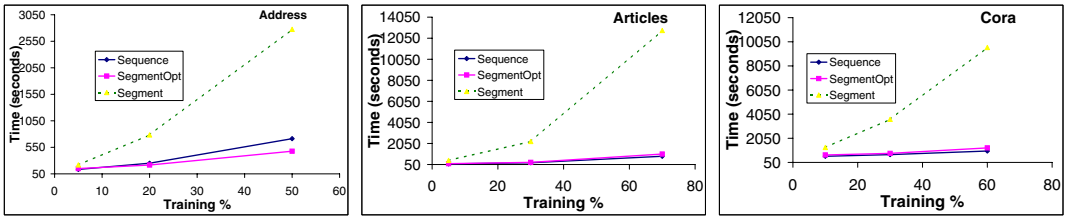

Figure 2: Running time against training set size for SequenceBCEU, Segment and SegmentOpt.

putation of all the terms without such optimization.

Algorithm 1 Alphas$(n, \psi_{i':\geq i}, \psi_{\leq i':i}, \psi_{\leq i':\geq i}, \psi_{i':i}, \theta_i)$

```
Initialize: α̂_1 = ⊬, A(0) = 0
for i = 1 ... n do
  Initialize θ_{<i+1:i} = ⊬, α_i = ⊬, α_i^P = ⊬
  for i' = i down to A(i − 1) + 1 do
    Get θ_{<i':i}, θ_{i':i} from θ_{<i'+1:i} and ψs
    α_i = α_i + α̂_{i'} θ_{i':i}
  end for
  Compute A(i) from ψs
  if A(i) > 0 then
    Get θ_{j:>i}, θ_{<j:>i} from θ_{<j:i} with j = A(i − 1) + 1,
    for j = A(i − 1) + 1 ... A(i) do
      α_i^P = α_i^P + α̂_j θ_{j:>i}
      Get θ_{j+1:>i}, θ_{<j+1:>i} from θ_{<j:>i} and ψs.
    end for
  end if
  if A(i − 1) > 0 then
    Get a_i, a_{>i} from a_{>i−1} and ψs.
    α_i = α_i + α_{i−1}^P a_i
    α_i^P = α_i^P + α_{i−1}^P a_{>i}
  end if
  α̂_{i+1} = α_i θ_{i+1}
end for
```

3.3 Complexity Analysis

We show that the *worst case* complexity of inference is $O(nm + H)$ where m is the maximum span of any potential and H is the total number of features expressed as potentials. Consider the computation of $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^P$ terms via Eq. 9. The maximum number of terms summed over in each of the equations is m, the maximum gap between end positions of any potential. This explains the $O(nm)$ part. We explain the $O(H)$ part by showing that the $\boldsymbol{\theta}$-s can be computed in such a way that if the same potential is applicable to k adjacent segments the amount of work done is a constant independent of k. For this we start from $i' = i$ and decrease i' and in each iteration compute $\boldsymbol{\theta}_{i':i}$ and $\boldsymbol{\theta}_{<i':i}$ from a previous computation of $\boldsymbol{\theta}_{<i'+1:i}$. We maintain the features in an efficient array-like structure such that for each $i' : i$ pair (there can be at most nm of these) and for each of the four possible kinds of potentials of the form $\boldsymbol{\psi}_{\leq i':\geq i}, \boldsymbol{\psi}_{\leq i':i}, \boldsymbol{\psi}_{i':\geq i}, \boldsymbol{\psi}_{i':i}$, we can retrieve all applicable features in $O(1)$ amortized time. Then, $\boldsymbol{\theta}_{i':i}$ can be computed from $\boldsymbol{\theta}_{<i'+1:i}$ by adding only the newly active features. We show how to compute the forward $\boldsymbol{\alpha}$ terms via efficient potential reuse in Algorithm 1. The computation of betas is analogous.

In contrast, for the original algorithm the complexity in the *average case* is $O(nL + G)$ where L is a hard limit on the maximum segment length and is expected to be greater than m and G is the total number of features fired over all segments. We show that $O(G) = O(L^2H)$ in the presence of token-level features. For example, consider a feature with the corresponding potential of the form $\boldsymbol{\psi}_{\leq s:\geq s+\ell-1}$. This feature would be fired $(L - \ell + 1)(L - \ell + 2)/2$ times since it overlaps with that many segments. If all features were at the word level with $\ell = 1$, then $O(G) = O(L^2H)$. In tasks like title/journal name extraction from citations where L is around 20 and several token-level features (like word and regular expression patterns indicators) are mixed with a few segment-level features (like match with a dictionary), this can lead to enormous savings as we see in the experimental section below. If we were to express complexity without involving the G and H terms and assumed that we need to perform $O(k)$ work to find potential of a k length segment, we get $O(nL^2)$ average case complexity for the old segmentation algorithm, which is reduced to $O(nm^2)$ worst case for the new algorithm.

The marginals $\boldsymbol{\mu}$ can be computed in $O(mn)$ time as follows. All features are first sorted in increasing order of their start boundary followed by a decreasing order of their end boundary. The computation of $\boldsymbol{\mu}$s is done in the same order so that no storage needs to be allocated for them and the compute cost of $\boldsymbol{\mu}$ is shared across all features with the same start and end boundary.

Empirically too we find the new inference algorithm to significantly reduce training time. In Fig. 2 we show that the running time for Segment increases sharply whereas SegmentOpt stays close to SequenceBCEUwith increasing training size on three different tasks. More details about the task and experiments can be found in Sarawagi[19].

4 Applications of Semi-CRFs

Ever since the work was first published in 2004, there have been many applications of the model. We list them below.

Integrating unstructured text into relational databases Consider an application where we need to automatically integrate unstructured text into an existing large multi-relational structured database. For example, given a structured database of publications, we need to integrate a new unstructured citation string by first extracting structured fields and then deduping with existing fields in the database. The demi-CRF model provides a convenient method of exploiting match with existing entries in the database as segment-level features, and then to jointly both extract the entities and dedupe it with an existing entity if it is a potential duplication. Mansuri and Sarawagi[16] presents the design of such a system.

Creating probabilistic databases from information extraction models Large text databases obtained by integrating unstructured data from multiple sources are central to many real-life applications, including citation databases like Citeseer, product comparison databases, and personal information management systems (PIM). A key step in the creation of such databases is the extraction of structured entities from unstructured sources. The semi-CRF model provides a sound probability distribution over extractions but is not easy to represent and query in a relational framework. One method of surmounting the problem of extraction errors is to require that each extracted entity be attached with confidence scores that correlate with the probability that the extracted entities are correct. Such probabilistic results can be stored in an imprecise data management system[1, 4–7, 21] for getting probabilistic answers. While this representation of uncertainty is natural and allows for simple query execution semantics, the number of extraction results can in the worst case be exponentially large. Gupta and Sarawagi[8] shows how a semi-CRF model can be converted into a mixture of multinomial distributions over the different fields extracted by the model.

Extracting tables from lists on the web In this application, our goal was to assemble a table from a few example rows by harnessing the huge corpus of information-rich but unstructured lists on the web. Consider a scenario where a user wishes to build a table of multi-attribute records belonging to a topic, say oil spills, starting with a few editor-picked seed records. We wish to construct this answer by aggregating the oil spill mentions present in various HTML lists on the Web. This requires us to segment the list records and map the segments to the query schema using a statistical model and consolidate the results from multiple lists into a unified merged table. Gupta and Sarawagi[9] shows a method of solving this task by first matching the seed set of examples to the HTML lists to create noisily labeled data, then training a semi-CRF based on segment-level regular expressions over HTML tags, exploiting overlap across lists to collect even more labeled data, and so on in a loop.

Recent applications of semi-CRFs More recently, semi-CRFs have been combined with deep learning that provides richer feature representations than earlier hand-crafted features. Some examples include part of speech tagging of character-level noisy text[10], named entity recognition on word level input[23, 25], hand writing and speech recognition[2].

5 Conclusion and Future Work

Semi-CRFs are a tractible extension of CRFs that offer much of the power of higher-order models without the associated computational cost. A major advantage of semi-CRFs is that they allow features which measure properties of segments, rather than individual elements. For applications like NER and gene finding[12], these features can be quite natural. In experiments on several NER problems, semi-CRFs generally outperform conventional CRFs, and when little training data is available, the differences are often dramatic.

The default forward–backward inference algorithm for inference in semi-CRFs is significantly slower than in CRFs. We also proposed techniques to make the running time of semi-Markov models comparable to that of CRFs by exploiting features shared across multiple segments.

Since its publication in 2004, the semi-CRF model has been adopted in many applications, including recent deep learning models for part of speech tagging, named entity recognition, hand writing recognition, and speech recognition. However, none of these recent implementations harness the overlapping forward–backward inference algorithm . A compelling direction of future work is to train deep learning models with this more efficient inference algorithm.

Publisher's Note

Received: 1 November 2018 Accepted: 24 January 2019
Published online: 20 March 2019

References

1. Barbar D, Garcia-Molina H, Porter D (1992) The management of probabilistic data. IEEE Trans Knowl Data Eng 4(5):487–502. https://doi.org/10.1109/69.166990
2. Beck E, Hannemann M, Dtsch P, Schlter R, Ney H (2018) Segmental encoder-decoder models for large vocabulary automatic speech recognition. Proc Interspeech 2018:766–770
3. Borthwick A, Sterling J, Agichtein E, Grishman R (1998) Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In: Sixth Workshop on Very Large Corpora. Association for Computational Linguistics, New Brunswick, New Jersey
4. Boulos J, Dalvi N, Mandhani B, Mathur S, Re C, Suciu D (2005) Mystiq: a system for finding more answers by using probabilities. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland. https://doi.org/10.1145/1066157.1066277
5. Dalvi NN, Suciu D (2004) Efficient query evaluation on probabilistic databases. In: Proceedings of the 30th VLDB Conference,Toronto, Canada, pp 864–875
6. Fuhr N (1990) A probabilistic framework for vague queries and imprecise information in databases. In: Proceedings of the sixteenth international conference on Very large databases. Morgan Kaufmann Publishers Inc., San Francisco, pp 696–707
7. Green TJ, Tannen V (2006) Models for incomplete and probabilistic information. IEEE Data Eng Bull 29(1)
8. Gupta R, Sarawagi S (2006) Curating probabilistic databases from information extraction models. In: VLDB
9. Gupta R, Sarawagi S (2009) Answering table augmentation queries from unstructured lists on the web. In: PVLDB
10. Kemos A, Adel H, Schtze H (2018) Neural semi-markov conditional random fields for robust character-based part-of-speech tagging. 1808.04208
11. Keshet J, Shalev-Shwartz S, Singer Y (2005) Phoneme alignment using large margin techniques. In: Workshop on the advances in structured learning for text and speech processing, NIPS
12. Krogh A (1998) Gene finding: putting the parts together. In: Bishop MJ (ed) Guide to human genome computing, 2nd edn. Academic Press, Cambridge, pp 261–274
13. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the International Conference on Machine Learning (ICML-2001), Williams
14. Liu DC, Nocedal J (1989) On the limited memory bfgs method for large-scale optimization. Math Programm 45:503–528
15. Malouf R (2002) A comparison of algorithms for maximum entropy parameter estimation. In: Proceedings of The sixth conference on natural language learning (CoNLL-2002), pp 49–55
16. Mansuri I, Sarawagi S (2006) A system for integrating unstructured data into relational databases. In: Proc. of the 22nd IEEE Int'l Conference on Data Engineering (ICDE)
17. McCallum A, Li W (2003) Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In: Proceedings of The Seventh Conference on Natural Language Learning (CoNLL-2003), Edmonton, Canada
18. McDonald R, Crammer K, Pereira F (2005) Flexible text segmentation with structured multilabel classification. In: HLT/EMNLP
19. Sarawagi S (2006) Efficient inference on sequence segmentation models. In: Proceedings of the 23rd International Conference on Machine Learning (ICML), Pittsburgh
20. Sarawagi S, Cohen WW (2004) Semi-markov conditional random fields for information extraction. In: NIPS
21. Sarma AD, Benjelloun O, Halevy A, Widom J (2006) Working models for uncertain data. In: ICDE
22. Sha F, Pereira F (2003) Shallow parsing with conditional random fields. In: Proceedings of HLT-NAACL
23. Ye ZX, Ling ZH (2018) Hybrid semi-markov crf for neural sequence labeling. In: ACL
24. Zhang T, Damerau F, Johnson D (2002) Text chunking based on a generalization of winnow. J Mach Learn Res 2:615–637
25. Zhuo J, Cao Y, Zhu J, Zhang B, Nie Z (2016) Segment-level sequence modeling using gated recursive semi-markov conditional random fields. In: ACL

Sunita Sarawagi researches in the fields of databases, data mining, and machine learning. Her current research interests are deep learning, graphical models and information extraction. She is institute chair professor at IIT Bombay. She got her PhD in databases from the University of California at Berkeley and a bachelor degree from IIT Kharagpur. Her past affiliations include visiting faculty at Google Research, Mountain view, CA, visiting faculty at CMU Pittsburg, and research staff member at IBM Almaden Research Center. She has several publications in databases and data mining, and several patents. She serves on the board of directors of ACM SIGKDD and VLDB foundation. She was program chair for the ACM SIGKDD 2008 conference, research track co-chair for the VLDB 2011 conference and has served as program committee member for SIGMOD, VLDB, SIGKDD, ICDE, and ICML conferences. She is/was on the editorial board of the ACM TODS, ACM TKDD, and FnT for machine-learning journals.