Efficient Inference on Sequence Segmentation Models

Sunita Sarawagi SUNITA@IITB.AC.IN
IIT Bombay

Abstract

Sequence segmentation is a flexible and highly accurate mechanism for modeling several applications. Inference on segmentation models involves dynamic programming computations that in the worst case can be cubic in the length of a sequence. In contrast, typical sequence labeling models require linear time. We remove this limitation of segmentation models vis-a-vis sequential models by designing a succinct representation of potentials common across overlapping segments. We exploit such potentials to design efficient inference algorithms that are both analytically shown to have a lower complexity and empirically found to be comparable to sequential models for typical extraction tasks.

1. Introduction

Sequence segmentation models form the core of several applications including speech segmentation on phoneme boundaries (Keshet et al., 2005), information extraction (Sarawagi & Cohen, 2004), named entity recognition (McDonald et al., 2005), syntactic chunking (DauméIII & Marcu, 2005), 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 a 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 (Borthwick et al., 1998), and in syntactic chunking tasks is the Begin-Inside-Outside (BIO) encoding of labels (Zhang et al., 2002). 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 4 and 8 words, the third or fourth token of the entity is a "-", and the last three tokens in the entity are digits.

Sequence segmentation models provide a direct and natural way of encapsulating all such entity features. Given an input sequence $\boldsymbol{x} = x_1 \dots x_n$, a segmentation $\boldsymbol{s}$ of $\boldsymbol{x}$ consists of a sequence of variable length segments $\boldsymbol{s} = \langle s_1, \dots, s_p \rangle$ 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 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.

The primary limitation of segmentation models compared to sequence models is the increased computational overheads of inference tasks (DauméIII & Marcu, 2005; Sarawagi & Cohen, 2004). The segmentation problem has been cast in a number of learning frameworks, including max-margin methods (McDonald et al., 2005) and CRF-based conditional likelihood methods (Sarawagi & Cohen, 2004). In all these different frameworks, there is a need to compute either the most likely segmentation as in cutting plane methods (Tsochantaridis et al., 2005) or various marginal probabilities as in likelihood methods and large margin methods optimized via exponentiated gradient algorithms (Bartlett et al., 2005). Both these inference problems are linear in the length n of the sequence for sequential labeling models, but can be quadratic (and is typically cube as we show in the paper) in n for segmentation models. A practical solution is to limit the maximum length of a segment by an a priori parameter L but even with such a choice, the training time for segmentation models has been observed to be 3–10 times worse than sequential models for extraction applications.

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

In this paper we seek to remove this limitation. The key insight we exploit is that most features are common across several overlapping segments. We propose a representation of features where such overlap can be compactly expressed. We then design inference algorithms where the running time is proportional to the number of features irrespective of how many segments a feature overlaps with. The algorithm pays the penalty of the segmentation models only when there are longer entity level features — when all the features are traditional token-level features the running time is the same as the standard Viterbi of sequence models. We achieve this without placing any hard limit on the length of a sequence. Empirical results show that with the new algorithm the running time for inference in segmentation models can be reduced by almost an order of magnitude. For typical information extraction models this makes the running time for training likelihood-based segmentation models comparable to that of sequential models while achieving higher accuracy because of the addition of a small, yet powerful set of entity-level features.

2. Graphical Models for Segmentation

We partition the potentials of a segmentation model into two types:

- 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 features of an exponential model as $\theta_{i:j}(y) = \exp(\sum_{k=1}^{K} w_k f_k(\boldsymbol{x}, i, j, y))$ where w_k denotes the weight parameter of feature f_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'.

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 $\prod_{j=1}^{p} \boldsymbol{\theta}_{t_j}(y_{j-1}, y_j)\theta_{t_j:u_j}(y_j)$. During inference we need to find the sum or max marginal of a segment or edge potential. These inference problems can be solved optimally in polynomial time using an extension of the forward/backward algorithm for sequence labeling. 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 segmentations of the sequence between 1 to i where the last segment has a label y.

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

where L, the maximum segment length is a parameter of the learning algorithm. 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 segmentations of the sequence from $i+1$ to n with the segment ending at i having label y.

These can be used to find the marginal for a segment potential $i' : i$ labeled y as:

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

where $Z(\boldsymbol{x}) = \sum_y \alpha_n(y)$.

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. Similarly, we can find the most likely segmentation by replacing the two outer sums by two max terms in Equation 1.

Empirically, for typical NER tasks, segmentations 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 next show how to address this shortcoming of segmentation models.

3. Efficient Inference Algorithms

A key insight we employ is that potentials for segmentations 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.

- $\boldsymbol{\psi}_{i':\ge i}$ denotes potentials shared over all segments starting at i' but ending anywhere after or at i.
- $\boldsymbol{\psi}_{\le i':i}$ denotes potentials shared over all segments ending at i but starting anywhere before or at i'.

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

We show that in almost all applications of segmentations such kinds of potentials with varying levels of overlap across segments is commonplace.

Named Entity Recognition NER tasks are typically encoded via a BCEU encoding of labels. We can express all such potentials along with arbitrary other entity features without having to commit to a fixed encoding of labels. We list some examples of such potentials:

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

Syntactic chunking The set of features proposed in (DauméIII & Marcu, 2005) for syntactic chunking via segmentation also contains a mix of the four types of segment potentials. We highlight some examples:

- The token at position i before the start of a chunk is "it": $\boldsymbol{\psi}_{i+1:\geq i+1}$.
- The second and third token of a chunk starting at $i-1$ are stop-words: $\boldsymbol{\psi}_{i-1:\geq i+1}$.
- The case pattern for a NP segment between tokens i and $i+2$ is three-caps:$\boldsymbol{\psi}_{i:i+2}$.

Speech segmentation based on phoneme boundaries The input here is a sequence of frames and the goal is to segment the frames along a fixed number of phoneme boundaries. This can be treated as a segmentation problem and as shown in (Keshet et al., 2005) contains a variety of potential types as follows:

- Distance features between frames $i-s$ and $i+s$ for $s = 1, 2, 3, 4$ for a phoneme segment starting at i: $\boldsymbol{\psi}_{i:\geq i+s}$
- The classification scores for the frames between i' and i being a phoneme: $\boldsymbol{\psi}_{i':i}$

Our goal next is to exploit the succinct form of potentials to speed up the message passing algorithms outlined in Section 2. 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 Equation 1 with only full segment potentials.

We first show how this is done for forward and backward messages in Section 3.1. We then show in Section 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 $\boldsymbol{\theta}_{i':i}$ denote the product of all potentials applicable to segment $i' : i$; this can be expressed as $\boldsymbol{\theta}_{i':i} = \prod\{\boldsymbol{\psi}_{uv} : u = i' \vee u = (\leq k), k \geq i', v = i \vee v = (\geq j), j \leq i\}$. For example, in Figure 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 $\boldsymbol{\psi}_{3:\geq 5}$ whereas the two edges between 4 and 7 denote the potential $\boldsymbol{\psi}_{\leq 4:\geq 7}$ for the arrowed edge and $\boldsymbol{\psi}_{4:7}$ for the plain edge. For segment $4:7$ the total potential $\theta_{4:7} = \boldsymbol{\psi}_{4:\geq 5}\boldsymbol{\psi}_{4:7}\boldsymbol{\psi}_{\leq 4:\geq 7}\boldsymbol{\psi}_{\leq 6:7}$

Equation 1 for computing α values with no L restriction can be written in matrix notation as:

$$\boldsymbol{\alpha}_i = \sum_{i' \leq i} (\boldsymbol{\alpha}_{i'-1}\boldsymbol{\theta}_{i'}) * \boldsymbol{\theta}_{i':i} \tag{2}$$

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

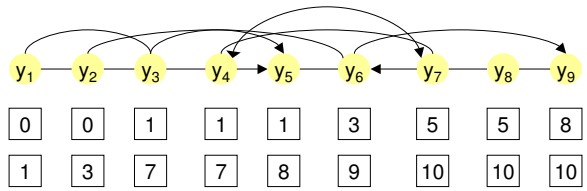

Figure 1. Potentials for a sequence of length 9. Potentials of the form $\boldsymbol{\psi}_{j:\geq i}$ are represented with edges having an arrow at i, potentials of the form $\boldsymbol{\psi}_{\leq j:i}$ are represented with edges having an arrow at j and so on. The first row of integers denotes $A(i)$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

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

$$\boldsymbol{\theta}_{i':i} = \boldsymbol{\theta}_{i':>i-1}\boldsymbol{\theta}_{i':(>i-1\rightarrow i)} \tag{3}$$

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

$$\boldsymbol{\theta}_{i':(>i-1\rightarrow i)} = \boldsymbol{\theta}_{A(i-1):(>i-1\rightarrow i)} \quad \text{if} \;\; i' \leq A(i-1) \tag{4}$$

Using equations 3 and 4, we can choose $a(i) = \boldsymbol{\theta}_{A(i-1):(>i-1\rightarrow i)}$ 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 2 can be rewritten as

$$\begin{aligned}\boldsymbol{\alpha}_i &= \sum_{i' \leq A(i-1)} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{i':>i-1}\boldsymbol{a}_i + \sum_{A(i-1)<i'\leq i} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{i':i} \\ &= \boldsymbol{\alpha}^P_{i-1}\boldsymbol{a}_i + \sum_{A(i-1)<i'\leq i} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{i':i} \end{aligned} \tag{5}$$

where $\boldsymbol{\alpha}^P_{i-1}$ denotes the sum of scores over all possible segmentations 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 Figure 1 we show how to compute $\boldsymbol{\alpha}_7$ given $A(6) = 3$.

$$\begin{aligned}\boldsymbol{\alpha}_7 &= \boldsymbol{\alpha}^P_6\boldsymbol{a}_7 + \hat{\boldsymbol{\alpha}}_4\boldsymbol{\theta}_{4:7} + \hat{\boldsymbol{\alpha}}_5\boldsymbol{\theta}_{5:7} + \hat{\boldsymbol{\alpha}}_6\boldsymbol{\theta}_{6:7} \quad (\text{see } Eq5) \\ &= \boldsymbol{\alpha}^P_6\boldsymbol{\psi}_{\leq 4:\geq 7}\boldsymbol{\psi}_{\leq 6:7} + \hat{\boldsymbol{\alpha}}_4\boldsymbol{\psi}_{4:\geq 5}\boldsymbol{\psi}_{4:7}\boldsymbol{\psi}_{\leq 4:\geq 7}\boldsymbol{\psi}_{\leq 6:7} + \\ &\quad \hat{\boldsymbol{\alpha}}_5\boldsymbol{\psi}_{\leq 6:7} + \hat{\boldsymbol{\alpha}}_6\boldsymbol{\psi}_{\leq 6:7}\end{aligned}$$

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,

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

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

$$\boldsymbol{\theta}_{i:i'} = \boldsymbol{\theta}_{<i+1\rightarrow i:i'}\boldsymbol{\theta}_{<i+1:i'} \tag{6}$$

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

$$\boldsymbol{\theta}_{<i+1\rightarrow i:i'} = \boldsymbol{\theta}_{<i+1\rightarrow i:B(i+1)} \quad \text{if } i' \geq B(i+1) \tag{7}$$

Using equations 6 and 7 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:

$$\begin{aligned}\boldsymbol{\beta}_i &= \boldsymbol{\theta}_{i+1}\Big(\sum_{B(i+2)>i'>i} \boldsymbol{\theta}_{i+1:i'}\boldsymbol{\beta}_{i'} + \sum_{i' \geq B(i+2)} \boldsymbol{b}_{i+1}\boldsymbol{\theta}_{<i+2:i'}\boldsymbol{\beta}_{i'}\Big) \\ &= \boldsymbol{\theta}_{i+1}\Big(\sum_{B(i+2)>i'>i} \boldsymbol{\theta}_{i+1:i'}\boldsymbol{\beta}_{i'} + \boldsymbol{b}_{i+1}\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{i+2}}\Big) \quad (8)\end{aligned}$$

where $\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{i}} = \sum_{i' \geq B(i)} \boldsymbol{\theta}_{<i:i'}\boldsymbol{\beta}_{i'}$, represents the sum over all segmentations $\boldsymbol{s}$ 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:

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

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:

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

The first case in the above equation follows from Equation 5 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: 4, 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:

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

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:

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

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.

$$\begin{aligned}\boldsymbol{\mu}_{\leq s:\geq e} &= \sum_{i' \leq s} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{<s+1 \rightarrow i':e} \sum_{i \geq e} \boldsymbol{\theta}_{<s+1:i}\boldsymbol{\beta}_i \quad (\textit{see Eq}: 8) \\ &= \sum_{i' \leq s} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{<s+1 \rightarrow i':e}\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s+1}} \\ &= \sum_{i' \leq s} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{\theta}_{<i'+1 \rightarrow i':e} \prod_{i' < j \leq s} \boldsymbol{\theta}_{<j+1 \rightarrow <j:e}\ \boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s+1}} \\ &= \sum_{i' \leq s} \hat{\boldsymbol{\alpha}}_{i'}\boldsymbol{b}_{i'} \prod_{i' < j \leq s} \boldsymbol{b}_{<j}\ \boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s+1}} \quad (\text{see Eq 7}) \\ &= ((\boldsymbol{\mu}_{\leq s-1:\geq B(s)}/\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s}})\boldsymbol{b}_{<s} + \hat{\boldsymbol{\alpha}}_s\boldsymbol{b}_s)\boldsymbol{\beta}^{\boldsymbol{P}}_{\boldsymbol{s+1}}\end{aligned}$$

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 computation of all the terms without such optimization.

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

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

Initialize: $\hat{\boldsymbol{\alpha}}_1 = \mathbb{1}$, $A(0) = 0$
for $i = 1 \ldots n$ **do**
  Initialize $\boldsymbol{\theta}_{<i+1:i} = \mathbb{0}, \boldsymbol{\alpha}_i = \mathbb{0}, \boldsymbol{\alpha}_i^P = \mathbb{0}$
  for $i' = i$ down to $A(i-1)+1$ **do**
    Get $\boldsymbol{\theta}_{<i':i}, \boldsymbol{\theta}_{i':i}$ from $\boldsymbol{\theta}_{<i'+1:i}$ and $\boldsymbol{\psi}$s
    $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i + \hat{\boldsymbol{\alpha}}_{i'} \boldsymbol{\theta}_{i':i}$
  Compute $A(i)$ from $\boldsymbol{\psi}$s
  if $A(i) > 0$ **then**
    Get $\boldsymbol{\theta}_{j:>i}, \boldsymbol{\theta}_{<j:>i}$ from $\boldsymbol{\theta}_{<j:i}$ with $j = A(i-1)+1$,
    for $j = A(i-1)+1 \ldots A(i)$ **do**
      $\boldsymbol{\alpha}_i^P = \boldsymbol{\alpha}_i^P + \hat{\boldsymbol{\alpha}}_j \boldsymbol{\theta}_{j:>i}$
      Get $\boldsymbol{\theta}_{j+1:>i}, \boldsymbol{\theta}_{<j+1:>i}$ from $\boldsymbol{\theta}_{<j:>i}$ and $\boldsymbol{\psi}$s.
  if $A(i-1) > 0$ **then**
    Get $\boldsymbol{a}_i, \boldsymbol{a}_{>i}$ from $\boldsymbol{a}_{>i-1}$ and $\boldsymbol{\psi}$s.
    $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i + \boldsymbol{\alpha}_{i-1}^P \boldsymbol{a}_i$
    $\boldsymbol{\alpha}_i^P = \boldsymbol{\alpha}_i^P + \boldsymbol{\alpha}_{i-1}^P \boldsymbol{a}_{>i}$
  $\hat{\boldsymbol{\alpha}}_{i+1} = \boldsymbol{\alpha}_i \boldsymbol{\theta}_{i+1}$

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 Equation 5. 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 $O(G) = O(L^2 H)$ 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 $O(G) = O(L^2 H)$. 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.

4. Experiments

We compare the optimized segmentation model (SegmentOpt) with the unoptimized model (Segment) and, for reference, also with the popular sequential labeling model with the begin-continue-end-unique (BCEU) encoding of labels (SequenceBCEU). We train each of these models to maximize the conditional likelihood objective regularized with a Gaussian prior on the parameters.

Datasets: We compared on three different extraction tasks spanning eight labels. The *Cora_Journals* dataset is the popular Cora citations dataset (Peng & McCallum, 2004) with two modifications: in the original Cora dataset, all authors are combined in a single label whereas we separate out individual authors, this makes the extraction problem harder but more meaningful and allows better exploitation of author name lexicons. We subset to include only journals (163 in all) to exploit journal and author names lexicons in the publicly available DBLP database. We extract title, authors and journal names. The *Address* corpus consists of 395 home addresses of students in a major university in a country with much less structure to their addresses than typical US addresses. We considered extraction of city names and state names from this corpus. For this dataset, we procured an external dictionary of city names and state names obtained from a postal database. The *Articles* dataset was created by starting from 400 journal entries over all the author's .bib files collected over several years. The unstructured records consisted of 100 citations obtained

Table 1. Comparing various methods on three IE tasks, with and without semi-Markov features. Column R is recall, P is precision and Time refers to the total training and testing time in seconds. SegmentOpt abbreviates SegmentOpt and Segment abbreviates Segment.

	Markov features (m=1)					Semi-Markov ($m = 7, 20, 20$)					
dataset	method	R	P	F1	Time	method	R	P	F1	Time	$\|G\|/\|H\|$
Address	SequenceBCEU	81.8	84.6	83.1	397	Segment	88.6	89.6	89.0	1130	8261
	SegmentOpt	85.5	87.9	86.5	292	SegmentOpt	88.4	89.9	89.1	342	2152
Articles	SequenceBCEU	81.7	84.8	82.9	220	Segment	85.5	86.4	85.8	2241	130826
	SegmentOpt	83.2	85.0	83.9	139	SegmentOpt	85.3	86.2	85.6	257	17015
Cora	SequenceBCEU	90.9	90.7	90.8	696	Segment	91.7	89.7	90.7	3623	193921
	SegmentOpt	91.4	89.9	90.6	557	SegmentOpt	91.6	89.7	90.6	800	21238

by searching Citeseer for citations of a random subset of authors appearing in the Bibtex database. The dataset is publicly available [2]. We extracted individual author names, title and journal names. The main difference between this and the Cora dataset is that the lexicon has a greater overlap for this dataset.

Features: We used indicator features for the word itself and various surface patterns (capturing capitalization, digit pattern and delimiters) at the word and one position to its left and right. These features apply on the start, end and in-between part of segments – to get features equivalent to those in SequenceBCEU. Transition features are used with a history size of 1. These are all word-level features. The segment-level features we added were length feature for each possible segment-length value and for each available external lexicon we added features corresponding to the highest match with whole entities in the lexicon. We used two similarity functions (TFIDF, and JaroWinkler) which are known to work well for name-matching in data integration tasks (Cohen et al., 2003). For the benefit of the sequence models we also added word level dictionary match features using the same similarity functions. Thus, we believe that an equivalent set of features are available to the sequence and segmentation models.

We measure F1 accuracy of classifying the entire field correctly. The numbers are averaged over four random selections of 30% of the data for training and the rest for testing.

In Table 1 we compare SequenceBCEU with SegmentOpt (without any segment-level features), and Segment with SegmentOpt both with segment level features of maximum length 7 for the address data, and 20 for the other two. The numbers are averaged over all extracted labels in the dataset. We observe that F1 improves from 83 to 89 for the address data, 84 to 86 for Articles, and remains unchanged at 91 for the Cora dataset. This supports the conclusions in earlier work that segmentation models are better suited for extraction tasks. The running time of Segment is a factor of 3–10 of SequenceBCEU. This shortcoming is removed with SegmentOpt where we are able to get the same accuracy advantage at the cost of only a factor 1–2 times higher running time compared to SequenceBCEU. Even with m of segment-level features as high as 20, the running time is comparable to the time of a sequence model. This gain is explained by the large difference in the sizes of features fired per sequence as shown in the last column. For example, the original set of 193,921 feature firings are represented equivalently as 21,238 feature firings (a factor of 10 reduction) with the succinct representation.

The improvement in running time achieved by SegmentOpt becomes more significant with increasing training fraction as we show in Figure 2 where running time for Segment increases sharply whereas SegmentOpt stays close to SequenceBCEU.

In Table 2 we illustrate the penalty of picking a wrong segment size L for Segment where we show F1 accuracy and running time with increasing segment size (parameter L for Segment and m for SegmentOpt). For a small segment size (10) in the Cora dataset accuracy of Segment is only 81 whereas with a segment size of 20 accuracy increases to 90.7, but this also doubles the running time. In contrast, for SegmentOpt accuracy remains steady around 90 and running time increases by only 15%.

Discussion In these experiments, the training objective was maximizing conditional likelihood; this requires marginal probabilities of potentials. We expect similar gains on max-margin methods trained via the exponentiated gradient optimization methods (Bartlett et al., 2005) since they require the same set of marginals. The training of max-margin methods via cutting plane algorithms (McDonald et al., 2005; Keshet et al., 2005; Tsochantaridis et al., 2005) that require finding the highest scoring segmentations will also show improvements, although we expect the gap to be smaller. Recently, DauméIII and Marcu (2005) propose a new structured learning framework where

[2] http://www.it.iitb.ac.in/s̃unita/data/personalBib.tar.gz

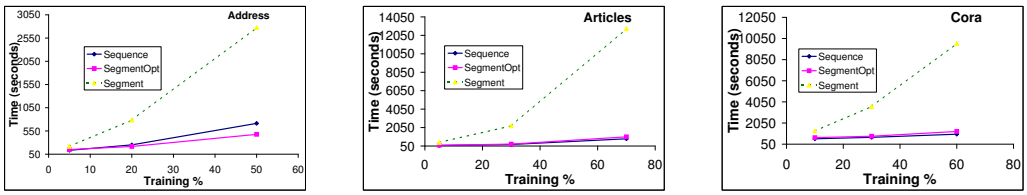

Figure 2. Running time against training set size for SequenceBCEU, Segment and SegmentOpt.

Table 2. Sensitiveness to segment size in the two semi-Markov models.

		Articles				Cora			
	Segment size →	10	15	20	25	10	15	20	25
F1	Segment	83.5	85.0	85.8	85.8	81.0	88.8	90.7	90.3
	SegmentOpt	85.5	85.7	85.6	85.6	90.8	90.1	90.6	90.3
Time(s)	Segment	1026	1810	2241	2698	1687	2771	3623	3813
	SegmentOpt	175	245	257	260	712	767	800	781

the search for the best segmentation is integrated with parameter training. Our main idea of defining potentials over a larger equivalence class of segments and the modified dynamic programming algorithm is also applicable in this setting.

Concluding remarks Segmentation of sequences provide a natural, flexible, and high-accuracy modeling of several applications. The main limitation to their adoption vis-a-vis sequential models has been increased computational cost of inference algorithms. We removed this limitation by designing a compact feature representation for the mix of token-level and segment-level features in typical segmentation tasks, and exploited these to design efficient inference algorithms. The algorithm pays the penalty of entity-level features only when needed and is comparable to Viterbi on sequential models when entity features are a small fraction of the total feature set. In addition, there is no a priori hard limit on the maximum length of a segment. In future we would like to generalize to feature values that are continuous functions of the distance from the segment boundaries, for example, a feature that captures a Gaussian distribution on the length of the segment. Another interesting future work is generalizing to 2D segmentation models.

Acknowledgements The work reported here was supported by research grants from Microsoft Research and an IBM Faculty award.

References

Bartlett, P. L., Collins, M., Taskar, B., & McAllester, D. (2005). Exponentiated gradient algorithms for large-margin structured classification. In L. K. Saul, Y. Weiss and L. Bottou (Eds.), *Advances in neural information processing systems 17*, 113–120. Cambridge, MA: MIT Press.

Borthwick, A., Sterling, J., Agichtein, E., & Grishman, R. (1998). Exploiting diverse knowledge sources via maximum entropy in named entity recognition. *Sixth Workshop on Very Large Corpora New Brunswick, New Jersey. Association for Computational Linguistics.*.

Cohen, W. W., Ravikumar, P., & Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*. To appear.

DauméIII, H., & Marcu, D. (2005). Learning as search optimization: approximate large margin methods for structured prediction. *ICML '05: Proceedings of the 22nd international conference on Machine learning* (pp. 169–176).

Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2005). Phoneme alignment using large margin techniques. *Workshop on the Advances in Structured Learning for Text and Speech Processing, NIPS*.

McDonald, R., Crammer, K., & Pereira, F. (2005). Flexible text segmentation with structured multilabel classification. *Human Language Technology Conference Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Peng, F., & McCallum, A. (2004). Accurate information extraction from research papers using conditional random fields. *HLT-NAACL* (pp. 329–336).

Sarawagi, S., & Cohen, W. W. (2004). Semi-markov conditional random fields for information extraction. *NIPS*.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR), 6(Sep)*, 1453–1484.

Zhang, T., Damerau, F., & Johnson, D. (2002). Text chunking based on a generalization of winnow. *J. Mach. Learn. Res.*, *2*, 615–637.