Sparse Higher Order Conditional Random Fields for improved sequence labeling

Xian Qian[†] QIANXIAN@FUDAN.EDU.CN
Xiaoqian Jiang [‡] XQJIANG@MIT.EDU
Qi Zhang[†] QI_ZHANG@FUDAN.EDU.CN
Xuanjing Huang[†] XJHUANG@FUDAN.EDU.CN
Lide Wu[†] LDWU@FUDAN.EDU.CN
[†]School of Computer Science, Fudan University, Shanghai, 200433, P.R.China
[‡]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.

Abstract

In real sequence labeling tasks, statistics of many higher order features are not sufficient due to the training data sparseness, very few of them are useful. We describe Sparse Higher Order Conditional Random Fields (SHO-CRFs), which are able to handle local features and sparse higher order features together using a novel tractable exact inference algorithm. Our main insight is that states and transitions with same potential functions can be grouped together, and inference is performed on the grouped states and transitions. Though the complexity is not polynomial, SHO-CRFs are still efficient in practice because of the feature sparseness. Experimental results on optical character recognition and Chinese organization name recognition show that with the same higher order feature set, SHO-CRFs significantly outperform previous approaches.

1. Introduction

In sequence labeling tasks, structured learning models owe a great part of their success to the ability in using local structured information, such as Conditional Random Fields (CRFs) (Lafferty et al., 2001), Averaged Perceptron (Collins, 2002a), Max Margin Markov Networks (Taskar et al., 2003), etc. However, these approaches are inefficient to cover long distance features due to high computational complexity.

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

Recent approaches attempting to capture non-local features can be divided into four classes. The first class employs approximate inference algorithms such as Loopy Belief Propagation, Gibbs sampling. Despite of their simplicity, approximate inference techniques are not guaranteed to converge to a good approximation. The second class uses reranking framework such as (Collins, 2002b). These approaches typically generate N best candidate predictions, then adopt a post processing model to rerank these candidates using non-local features. The main drawback of these methods is that the effectiveness of post processing model is restricted by the number of candidates. The third class chooses Semi-Markov chain as the graphic model, such as Semi-Markov CRFs (Sarawagi & Cohen, 2004). Though the inference is exact and efficient, it can only deal with segment-based higher order features. The last class formulates the labeling task as a linear programming(LP) problem with some relaxations (Roth & tau Yih, 2005), so higher order features can be represented as linear constraints. For many higher order features, such inference is still approximate.

Different from the approaches mentioned above, we want to handle local and non-local features together while keeping the inference exact and tractable with some reasonable assumptions on non-local features. Our motivation is that in real applications, statistics of higher order features are not sufficient due to the training data sparseness, most of them may be useless. For example, in the optical character recognition(OCR) task, many higher order transitions are meaningless, such as "aaaa", "ijklmn", only very few of them are helpful, such as "tion", "ment". So in this sense, higher order features are sparse in terms of their contribution.

We propose Sparse Higher Order Conditional Ran-

dom Fields(SHO-CRFs) which can handle local and sparse higher order features together using a novel tractable exact inference algorithm. Though at the worst case, the complexity is still not polynomial, SHO-CRFs is quite efficient in practice due to feature sparseness. Experiments on optical character recognition(OCR) and Chinese organization name recognition tasks demonstrate our technique, SHO-CRFs significantly outperform conventional CRFs with the help of sparse higher order features, and outperform the candidate reranking approach with the same higher order features.

The paper is structured as follows: in section 2, we give the definition of configurations; in section 3, we describe our new inference algorithm using configuration graph; in section 4, we analyze the complexity of SHO-CRFs; experimental results are shown in section 5; we conclude the work in section 6.

2. Features and configurations

For probabilistic graphical models, the task of sequence labeling is to learn a conditional distribution $p(\mathbf{y}|\mathbf{x})$, where $\mathbf{x} = x_1 \dots x_l$ is the observed node sequence to be labeled, $\mathbf{y} = y_1 y_2 \dots y_l$ is the label sequence. Each component y_t is assumed to range over a finite label alphabet. For example, in named entity recognition(NER) task, we want to extract person names. Here $\mathbf{x}$ might be a sequence of words, and $\mathbf{y}$ might be a sequence in $\{B, I, O\}^{|\mathbf{x}|}$, where $y_t = B$ indicates "word x_t is the beginning of a person name", $y_t = I$ indicates "word x_t is inside a person name, but not the beginning" and $y_t = O$ indicates other cases.

CRFs are undirected graphic models depicted by Markov network distribution:

$$p(\mathbf{y}|\mathbf{x}) \quad \propto \quad \prod_{t=1}^{l} \phi(\mathbf{x}, \mathbf{y}, t)$$

where $\phi(\mathbf{x}, \mathbf{y}, t)$ is a real-valued potential function at position t, which has the form:

$$\phi(\mathbf{x}, \mathbf{y}, t) = \exp\left(\sum_i w_i f_i(\mathbf{x}, \mathbf{y}_{s:t})\right)$$

where w_i is the parameter to be estimated from the training data, f_i is a real valued feature function, or features for short, which is given and fixed. In this paper, for simplicity, we will focus on the case of binary features. However, our results extend easily to the general real valued case. We call a binary feature is fired if its value is true. $\mathbf{y}_{s:t} = y_s y_{s+1} \dots y_t$ is a label subsequence affected by f_i, $t - s$ is the order of f_i.

Formally, each feature can be factorized into two parts:

$$f(\mathbf{x}, \mathbf{y}_{s:t}) = b(\mathbf{x}, t) I_{\mathbf{Z}}(\mathbf{y}_{s:t})$$

Both parts are binary functions, $b(\mathbf{x}, t)$ indicates whether the observation satisfies certain characteristics at position t, $\mathbf{Z}$ is a set of label subsequences that are specified by f. $I_{\mathbf{Z}}(\mathbf{y}_{s:t})$ indicates whether $\mathbf{y}_{s:t} \in \mathbf{Z}$.

For example, we define a feature which is fired if the word subsequence lies between "professor" and "said" is recognized as a person name. Consider the sentence $\mathbf{x}$ ="Professor Abdul Rahman Haj Yihye said ...", we have

$$f_1(\mathbf{x}, \mathbf{y}_{2:5}) = b(\mathbf{x}, 5) I_{\{BIII\}}(\mathbf{y}_{2:5})$$

where

$$b(\mathbf{x}, 5) = \begin{cases} 1 & \text{if the 1 st word is "professor}'' \\ & \text{and the next word is "said}'' \\ 0 & \text{otherwise} \end{cases}$$

Another example is the feature used in skip chain CRFs, which is fired if a pair of same capitalized words have similar label. Suppose $\mathbf{x}$ ="Speaker John Smith ... Professor Smith will ...", "Smith" appears at position 3 and 100. Let $U = \{B, I, O\}$ denote the full label set, $\mathbf{U}_{4:99} = U \times \dots \times U$, and $\mathbf{Z} = \{B, I\} \times \mathbf{U}_{4:99} \times \{B, I\}$, we have,

$$f_2(\mathbf{x}, \mathbf{y}_{3:100}) = b(\mathbf{x}, 100) I_{\mathbf{Z}}(\mathbf{y}_{3:100})$$

where

$$b(\mathbf{x}, 100) = \begin{cases} 1 & \text{if the 3rd and 100 th words are} \\ & \text{the same and capitalized} \\ 0 & \text{otherwise} \end{cases}$$

Such feature is fired only if both "Smith" are labeled as a part of person name.

For a fired feature f at position t, if its corresponding $\mathbf{Z}$ yields the form $\mathbf{Z}_{s:t} = Z_s \times Z_{s+1} \times \dots \times Z_t$, where $Z_i \subseteq U, s \le i \le t$, such $\mathbf{Z}$ is called the **configuration** of f. Both examples mentioned above are configurations. However, for example, $\mathbf{Z} = \{BI, IB\}$ is not a configuration, in this case, we treat it as union of two configurations.

The potential function of a configuration is defined as:

$$\phi(\mathbf{Z}_{s:t}) = \exp\left(w f(\mathbf{x}, \mathbf{y}_{s:t})\right), \quad \text{for any } \mathbf{y}_{s:t} \sim \mathbf{Z}_{s:t}$$

where $\mathbf{y}_{p:q} \sim \mathbf{Z}_{s:t} (p \le s \le t \le q)$ indicates that subsequence $\mathbf{y}_{p:q}$ satisfies $\mathbf{y}_{s:t} \in \mathbf{Z}_{s:t}$.

3. Inference

3.1. Task

We describe our inference algorithm for training, which can be applied for decoding with a slight modification. In training stage, CRFs learn $\mathbf{w} = [w_1, \dots, w_M]^T$ from training data $\mathbf{X} = \{(\mathbf{x}^{(1)}, \tilde{\mathbf{y}}^{(1)}), (\mathbf{x}^{(2)}, \tilde{\mathbf{y}}^{(2)}) \dots\}$:

$$\min_{\mathbf{w}} \mathcal{O}(\mathbf{w}) = -\sum_i \log p(\tilde{\mathbf{y}}^{(i)}|\mathbf{x}^{(i)})$$

where $\tilde{\mathbf{y}}^{(i)}$ is the gold standard label sequence of $\mathbf{x}^{(i)}$, M is the feature number.

The goal of inference is to calculate $\mathcal{O}(\mathbf{w})$ and $\frac{\partial \mathcal{O}}{\partial \mathbf{w}}$. It is not difficult to obtain that

$$\frac{\partial \mathcal{O}}{\partial w_j} = \sum_i \sum_t \Big(\gamma(\mathbf{Z}_{i,t}) - f_j(\mathbf{x}^{(i)}, \tilde{\mathbf{y}}^{(i)}_{s:t})\Big)$$

where $\mathbf{Z}_{i,t}$ is the configuration of f_j at position t of the i^{th} training sample, and $\gamma(\mathbf{Z}_{i,t})$ is the marginal probability:

$$\gamma(\mathbf{Z}_{i,t}) = \sum_{\mathbf{y} \sim \mathbf{Z}_{i,t}} p(\mathbf{y}|\mathbf{x}^{(i)})$$

The inference task is to compute $\gamma(\mathbf{Z}_{i,t})$.

3.2. The configuration graph

Given a sequence, we could represent its configurations by a configuration graph. For example, suppose there are two configurations, $\mathbf{A}_{1:3} = \{B, I\} \times \{I\} \times \{B, I\}$ and $\mathbf{B}_{2:4} = \{I, O\} \times \{I, O\} \times \{B\}$, the corresponding configuration graph is shown in Figure 1, each configuration is represented by a box.

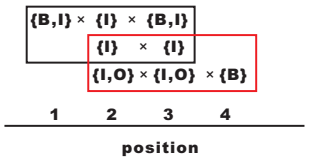

Figure 1. A configuration graph

Since $\gamma(\mathbf{Z}_{r:t}) = \sum_{\mathbf{y}_{r:t} \sim \mathbf{Z}_{r:t}} \gamma(\mathbf{y}_{r:t}) = \sum_{\mathbf{y}_{r-1:t} \sim U \times \mathbf{Z}_{r:t}} \gamma(\mathbf{y}_{r-1:t}) = \gamma(\mathbf{Z}'_{r-1:t})$, so a configuration could be extended to a wider range while keeping its marginal probability. Let r_{min} denote the leftmost position of the configurations at t, then all the configurations at t could be extended to $r_{min} : t$, so that they have same range. In the rest of the paper, we assume that such extension has been done for all configurations.

The extension operation will be frequently used in the rest of the paper, given a set of label subsequence, $\mathbf{A}_{s:t} = \{\mathbf{y}_{s:t}\}$ and new range from p to q, the extension is defined as:

$$E_{p:q}(\mathbf{A}_{s:t}) = \begin{cases} \mathbf{A}_{p:q} & s \le p \le q \le t \\ \mathbf{U}_{p:s-1} \times \mathbf{A}_{s:q} & p < s \le q \le t \\ \mathbf{A}_{p:t} \times \mathbf{U}_{t+1:q} & s \le p \le t < q \\ \mathbf{U}_{p:s-1} \times \mathbf{A}_{s:t} \times \mathbf{U}_{t+1:q} & p < s \le t < q \\ \mathbf{U}_{p:q} & \text{otherwise} \end{cases}$$

For configuration $\mathbf{Z}_{r:t}$, we wish to compute its marginal probability using

$$\gamma(\mathbf{Z}_{r:t}) = \frac{1}{z(\mathbf{x})} \sum_{\mathbf{y}_{r:t} \sim \mathbf{Z}_{r:t}} \alpha(\mathbf{x}, \mathbf{y}_{r:t}) \beta(\mathbf{x}, \mathbf{y}_{r:t}) \quad (1)$$

where $z(\mathbf{x})$ is the partition function,

$$\alpha(\mathbf{x}, \mathbf{y}_{r:t}) = \sum_{\mathbf{y}'_{1:t} \sim \{\mathbf{y}_{r:t}\}} \prod_{k=1}^{t} \phi(\mathbf{x}, \mathbf{y}'_{1:t}, k)$$

$$\beta(\mathbf{x}, \mathbf{y}_{r:t}) = \sum_{\mathbf{y}'_{r:l} \sim \{\mathbf{y}_{r:t}\}} \prod_{k=t+1}^{l} \phi(\mathbf{x}, \mathbf{y}'_{r:l}, k)$$

Intuitively, if we treat $\mathbf{y}$ as a flow that flows from left to right, $\alpha(\mathbf{x}, \mathbf{y}_{r:t})$ denotes the potential functions that have been obtained by flows end with $\mathbf{y}_{r:t}$, and $\beta(\mathbf{x}, \mathbf{y}_{r:t})$ denotes the potential functions that will be obtained by flows begin with $\mathbf{y}_{r:t}$.

However, formulation (1) is incorrect, consider the example in Figure 2, where $\mathbf{A}, \mathbf{B}$ are configurations at $t, t+1$ respectively, $\mathbf{y_1}_{r:t} = \mathbf{y_2}_{r:t} \in \mathbf{A}$. We could not compute $\beta(\mathbf{y_1}_{r:t})$, since $\beta(\mathbf{y_1}) \neq \beta(\mathbf{y_2})$, the reason is that, $\beta(\mathbf{y_1})$ excludes $\phi(\mathbf{B})$, while $\beta(\mathbf{y_2})$ does not. Hence we don't know whether $\beta(\mathbf{y_1}_{r:t}) = \beta(\mathbf{y_1})$ or $\beta(\mathbf{y_2})$.

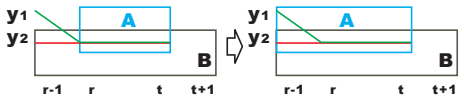

Figure 2. Error example of formulation (1) and correction

However, we could rectify this formulation by extending $\mathbf{A}_{r:t}$ to $\mathbf{A}_{r-1:t}$, so that $\mathbf{y_1}$ and $\mathbf{y_2}$ are two different elements of $\mathbf{A}_{r-1:t}$. Similarly, for configuration $\mathbf{C}_{p:t+2}$, if $p < r$, such extension is also required.

Therefore, we should consider the leftmost position (denoted as s) of configurations at $t+1, t+2, \ldots$ that overlaps $\mathbf{U}_{r:t}$, and extends configurations $\mathbf{Z}_{\mathbf{i}r:t}$ to $\mathbf{Z}_{\mathbf{i}s:t}$ if $s < r$. In the following, we assume that such extension has been done for all configurations, i.e., $r \leq s$.

In fact, the β value is unique for $\mathbf{y}_{s:t}$, even $s > r$, so we use the following equation for inference:

$$\gamma(\mathbf{Z}_{r:t}) = \frac{1}{z(\mathbf{x})} \sum_{\mathbf{y}_{r:t} \sim \mathbf{Z}_{r:t}} \alpha(\mathbf{x}, \mathbf{y}_{r:t}) \beta(\mathbf{x}, \mathbf{y}_{s:t}) \tag{2}$$

Conventional forward backward algorithm considers all $\mathbf{y}_{r:t} \sim \mathbf{U}_{r:t}$ to calculate $z(\mathbf{x})$ and $\gamma(\mathbf{Z})$, hence the complexity is exponential in the number of labels. However, if we could split $\mathbf{U}_{s:t}$ into several parts, so that all elements in one part share a common β value, then the complexity is reduced.

3.3. State partition

To derive a common β, we consider a simple case, $\mathbf{U}_{u:v}$ and one configuration $\mathbf{B}_{p:q}$, $q > v$ are given. If $p > v$, $\mathbf{B}_{p:q}$ and $\mathbf{U}_{u:v}$ are disjoint, then all $\mathbf{y}_{u:v} \sim \mathbf{U}_{u:v}$ share a common β: $\beta(\mathbf{y}_{u:v}) = \phi(\mathbf{B})|\mathbf{B}_{v+1:l}| + |\overline{\mathbf{B}_{v+1:l}}|$, where $\mathbf{B}_{v+1:l} = E_{v+1:l}(\mathbf{B}_{p:q})$. Hence no split needed.

If $p \leq v$, We have,

$$\beta(\mathbf{y}_{u:v}) = \begin{cases} \phi(\mathbf{B})|\mathbf{B}_{v+1:l}| + |\overline{\mathbf{B}_{v+1:l}}| & \mathbf{y}_{u:v} \sim \mathbf{B}_{u:v} \\ |\mathbf{U}_{v+1:l}| & \text{otherwise} \end{cases}$$

where $\mathbf{B}_{u:v} = E_{u:v}(\mathbf{B}_{p:q})$, as shown in Figure 3. Hence, we obtain the partition: $\{\mathbf{B}_{u:v}, \overline{\mathbf{B}_{u:v}}\}$, all members in one part share a common β.

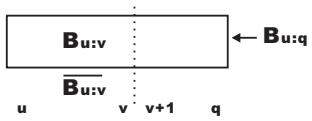

Figure 3. Split $\mathbf{U}_{u:v}$ so that all members in one part share a common β.

Generally, consider $\mathbf{U}_{s:t}$ mentioned in the previous sunsection, let $S = \{\mathbf{Z}_{\mathbf{i}p:q}\}$ denote the set of configurations at $t+1, t+2, \ldots$ that overlap $\mathbf{U}_{s:t}$, we extend each of them from $\mathbf{Z}_{\mathbf{i}p:q}$ to $\mathbf{Z}_{\mathbf{i}s:t}$, then derive a partition of $\mathbf{U}_{s:t}$, so that all $\mathbf{y}_{s:t}$ in one part share a common β:

$$\mathcal{P}(S) = \{\mathbf{A}_{\mathbf{k}s:t} | \mathbf{A}_{\mathbf{k}s:t} \neq \emptyset, k = 1, \ldots, 2^{|S|}\} \tag{3}$$

where $\mathbf{A}_{\mathbf{k}s:t} = \bigcap_{i=1}^{|S|} \mathbf{A}_{\mathbf{k}\mathbf{i}s:t}$, $\mathbf{A}_{\mathbf{k}\mathbf{i}s:t} = \mathbf{Z}_{\mathbf{i}s:t}$ or $\overline{\mathbf{Z}_{\mathbf{i}s:t}}$.

This partition is called the state partition of $\mathbf{U}_{s:t}$, denoted as $\pi_{s:t}$, each part $\mathbf{A}_{\mathbf{k}s:t} \in \pi_{s:t}$ is called a grouped state. The common β value of $\mathbf{A}_{\mathbf{k}s:t}$ is denoted by $\beta(\mathbf{A}_{\mathbf{k}s:t})$. $\mathcal{P}(S)$ is called the derived partition of configuration set S.

3.4. Transition partition

The next problem is to calculate the β value. First, we show two propositions:

Proposition 1

For any $\mathbf{A}_{s:t} \in \pi_{s:t}$ and any $y \in U$, we have, all $\mathbf{y}_{s:t+1} \sim \mathbf{A}_{s:t} \times \{y\}$ share a common potential function $\phi(\mathbf{x}, \mathbf{y}_{s:t+1}, t+1)$ at $t+1$.

Proof Let $S = \{\mathbf{Z}_{\mathbf{i}r:t+1}\}$ denote the set of configurations at $t+1$, and $T = \{\mathbf{Y}_{\mathbf{i}s:t}\}$,where $\mathbf{Y}_{\mathbf{i}s:t} = E_{s:t}(\mathbf{Z}_{\mathbf{i}r:t+1})$, for its derived partition $\mathcal{P}(T)$, each part can be represented by $\mathbf{C}_{s:t} = \left(\bigcap_{i \in I} \mathbf{Y}_{\mathbf{i}s:t}\right) \bigcap \left(\bigcap_{j \in T-I} \overline{\mathbf{Y}_{\mathbf{j}s:t}}\right)$, so $\mathbf{C}_{s:t} \times \{y\} = \left(\bigcap_{i \in I} \mathbf{Y}_{\mathbf{i}s:t} \times \{y\}\right) \bigcap \left(\bigcap_{j \in T-I} \overline{\mathbf{Y}_{\mathbf{j}s:t}} \times \{y\}\right)$. Hence for any $\mathbf{y}_{s:t+1} \sim \mathbf{C}_{s:t} \times \{y\}$, its potential function $\phi(\mathbf{x}, \mathbf{y}_{s:t+1}, t+1) = \prod_{i \in I, y \in Z_{i_{t+1}}} \phi(\mathbf{Z}_{\mathbf{i}})$. Since $\pi_{s:t}$ is refinement of $\mathcal{P}(T)$, so proposition 1 holds. □

Proposition 2

For any $\mathbf{A}_{s:t} \in \pi_{s:t}$, $\mathbf{B}_{r:t+1} \in \pi_{r:t+1}$, there exists an $Y_{t+1} \subseteq U$, so that $\mathbf{A}_{s:t} \times Y_{t+1} \subseteq \mathbf{U}_{s:r-1} \times \mathbf{B}_{r:t+1}$, and $\mathbf{A}_{s:t} \times \overline{Y_{t+1}} \bigcap \mathbf{U}_{s:r-1} \times \mathbf{B}_{r:t+1} = \emptyset$.

Proof is shown in Appendix A, which also gives such Y_{t+1}. An intuitive illustration is shown in the left part of Figure 4.

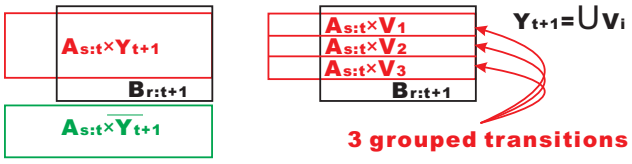

Figure 4. Left: Proposition 2, Right: 3 grouped transitions between grouped states $\mathbf{A}_{s:t}$ and $\mathbf{B}_{r:t+1}$

All $\mathbf{y}_{s:t+1} \sim \mathbf{A}_{s:t} \times Y_{t+1}$ share a common β. If we consider all $\mathbf{B}_{\mathbf{i}r:t+1} \in \pi_{r:t+1}$, then the set $\{\mathbf{A}_{s:t} \times Y_i | \mathbf{A}_{s:t} \times Y_i \subseteq \mathbf{U}_{s:r-1} \times \mathbf{B}_{\mathbf{i}r:t+1}, \mathbf{B}_{\mathbf{i}r:t+1} \in \pi_{r:t+1}\}$ is a partition of $\mathbf{A}_{s:t} \times U$, and the union of partitions of $\mathbf{A}_{\mathbf{i}s:t} \times U$ over $\mathbf{A}_{\mathbf{i}s:t} \in \pi_{s:t}$ is a partition of $\mathbf{U}_{s:t+1}$. According to proposition 1, all $\mathbf{y}_{s:t+1} \sim \mathbf{A}_{s:t} \times \{y\}$ share a common potential function , so we derived a partition of $\mathbf{A}_{s:t} \times Y_i = \bigcup_j \mathbf{A}_{s:t} \times V_j$ so that all $\mathbf{y}_{s:t+1}$ in one part $\mathbf{A}_{s:t} \times V_j$ share common β and ϕ values (denoted by $\phi(\mathbf{A}_{s:t} \times V_j)$). The union of such partitions over all $\{\mathbf{A}_{s:t} \times Y_i\}$ is a partition of $\mathbf{U}_{s:t+1}$, which is called

transition partition, denoted as $\varpi_{s:t+1}$. Each member $\mathbf{A}_{s:t} \times V_j$ is called a grouped transition. An example is shown in the right part of Figure 4.

We could compute $\beta(\mathbf{A}_{s:t})$ recursively by enumerating all its linked grouped transitions $\{\mathbf{A}_{s:t} \times V_{t+1}\}$:

$$\begin{aligned} & \beta(\mathbf{A}_{s:t}) \\ = & \sum_{\mathbf{A}_{s:t} \times V_{t+1}} \beta(\mathbf{B}_{V_{t+1}}) |V_{t+1}| \phi(\mathbf{A}_{s:t} \times V_{t+1}) \end{aligned}$$

where $\mathbf{B}_{V_{t+1}}$ is the grouped state $\mathbf{B}_{r:t+1}$ that satisfies $\mathbf{U}_{s:r-1} \times \mathbf{B}_{r:t+1} \supseteq \mathbf{A}_{s:t} \times V_{t+1}$.

3.5. The extended forward backward algorithm

We could build a trellis in which grouped states are represented by nodes, and edges between nodes denote grouped transition sets between two grouped states. If the transition set is empty, the corresponding edge does not exist. For instance, consider the first feature in section 2, suppose state features $f_i(y_t, \mathbf{x}, t)$ and first order transition features $g_i(y_{t-1}, y_t, \mathbf{x}, t)$ are used additionally, the trellis is shown in Figure 5.

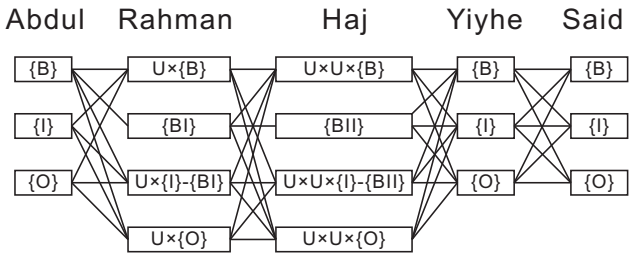

Figure 5. Trellis for the first feature example in section 2 with additional features $f_i(y_t, \mathbf{x}, t)$ and $g_i(y_{t-1}, y_t, \mathbf{x}, t)$

Recall that we use Equation (2) to compute marginal probability of $\mathbf{Z}_{r:t}$, we have $\mathbf{Z}_{r:t} = \bigcup_i \mathbf{T}_{\mathbf{i}r:t}$ where $\mathbf{T}_{\mathbf{i}r:t} \in \varpi_{r:t}$, and $\mathbf{T}_{\mathbf{i}r:t}$ has $\phi(\mathbf{Z}_{r:t})$. So the problem is to compute $\gamma(\mathbf{T}_{\mathbf{i}r:t})$.

Suppose grouped transition $\mathbf{T}_{s:t+1} = \mathbf{A}_{s:t} \times V_{t+1}$ links grouped states $\mathbf{A}_{s:t}$ and $\mathbf{B}_{r:t+1}$, we have,

$$\begin{aligned} & \gamma(\mathbf{T}_{s:t+1}) \\ \propto & \sum_{\mathbf{y}_{s:t+1} \sim \mathbf{T}_{s:t+1}} \alpha(\mathbf{y}_{s:t+1}) \beta(\mathbf{y}_{r:t+1}) \\ = & \beta(\mathbf{B}_{r:t+1}) \sum_{\mathbf{y}_{s:t+1} \sim \mathbf{T}_{s:t+1}} \alpha(\mathbf{y}_{s:t+1}) \\ = & \beta(\mathbf{B}_{r:t+1}) \phi(\mathbf{T}_{s:t+1}) |V_{t+1}| \sum_{\mathbf{y}_{s:t} \sim \mathbf{A}_{s:t}} \alpha(\mathbf{y}_{s:t}) \\ = & \beta(\mathbf{B}_{r:t+1}) \phi(\mathbf{T}_{s:t+1}) |V_{t+1}| \alpha(\mathbf{A}_{s:t}) \end{aligned}$$

where $\alpha(\mathbf{A}_{s:t}) = \sum_{\mathbf{y}_{s:t} \sim \mathbf{A}_{s:t}} \alpha(\mathbf{y}_{s:t})$, which could be calculated recursively:

$$\begin{aligned} & \alpha(\mathbf{B}_{r:t+1}) \\ = & \sum_{\substack{\mathbf{A}_{s:t} \times V_{t+1} \\ \subseteq \mathbf{U}_{s:r-1} \times \mathbf{B}_{r:t+1}}} \alpha(\mathbf{A}_{s:t}) |V_{t+1}| \phi(\mathbf{A}_{s:t} \times V_{t+1}) \end{aligned}$$

And $z(\mathbf{x}) = \sum_{\mathbf{A}_{s:l} \in \pi_{s:l}} \alpha(\mathbf{A}_{s:l})$.

4. Complexity analysis

In SHO-CRFs training, we have to derive state partitions and transition partitions of each sequence, a trellis like Figure 5 is built. Given extended configuration sets $A_t = \{E_{s:t}(\mathbf{Z}_{\mathbf{i}s_i:t_i})\}_{s_i \le t < t_i}$, and the configuration set at $t+1$ (denoted by B_{t+1}), the complexity to derive $\pi_{s:t}$ is $|\pi_{s:t}||A_t| \le |\varpi_{s:t+1}||A_t|$, the complexity to derive $\varpi_{s:t+1}$ is $|\varpi_{s:t+1}||B_{t+1}|$. In the worse case, $|\varpi_{s:t+1}|$ is exponential in label size when feature represents no sparsity at all. However in most real world scenarios, the size of $|\varpi_{s:t+1}|$ is usually limited, so building such a trellis is tractable. In the second step, the complexity of extended forward-backward algorithm is the sum of $|\varpi_{s:t}|$ over $\mathbf{x}, t$.

The ratio $\kappa(t) = \frac{|\varpi_{s:t}|}{|U|^{t-s+1}}$ is called the sparsity at t, $|\varpi_{s:t}|$, $|U|^{t-s+1}$ are the complexities of SHO-CRFs and conventional CRFs respectively. For example, the sparsity of the 4th word "Yiyhe" in Figure 5 is $\kappa(4) = 12/3^4$.

5. Experiments

5.1. Optical Character Recognition(OCR)

We conduct two experiments to demonstrate our method. The first is Optical Character Recognition (OCR) task. For comparison, we use the same data and settings as Ben Taskar's work(Taskar et al., 2003), where 6100 handwritten words were divided into 10 folds of 600 training and 5500 testing examples, averaged character accuracy over the 10 folds are reported as evaluation metric.

We compare our algorithm to four state of the art algorithms: traditional CRFs (Lafferty et al., 2001), M3Ns (Taskar et al., 2003), candidate reranking (Collins, 2002b) and structured Hidden Markov Model (SHMM) (Galassi et al., 2007), a variant of Hierarchical Hidden Markov Model that provides high level knowledge abstraction. For traditional CRFs, pixel features and first order transition features are used. The results of M3Ns are directly copied from Ben Taskar's paper. For candidate reranking and SHO-CRFs, we use the affixes as additional higher order features. We down-

Table 1. Distribution of Affixes

LENGTH	1	2	3	4	5	6	7	8
NUMBER	45	236	216	94	81	38	20	15

load the word list from the web[1], and get the prefixes and suffixes of length 1-8 with frequency higher than 100,100,100,100,70,70,70,50 respectively. Different thresholds are used for affixes of different length because frequencies of short affixes are generally higher than long affixes, using a single threshold will add noisy short affixes or miss important long affixes. The distribution of affixes of different length are shown in table 1.

An affix feature is defined as true if and only if the first or last several predicted characters form the corresponding affix. For candidate reranking, we use traditional CRFs to generate 100 candidates, then use averaged perceptron to rerank with affix features. For SHO-CRFs, local features and affix features are used simultaneously, the order of affix feature is one less than the length of corresponding affix, so SHO-CRF is a 7th order CRF. For SHMM, we use three blocks, one emits general observation and two affix (suffix and prefix) feature specific observation respectively. The parameters of every Gaussian mixture (3 Gaussians/State in our implementation) of each state of the HMM are estimated through Expectation Maximization.

Comparison results are shown in table 2. SHO-CRFs achieve the highest published accuracy. However, our work is orthogonal to kernel based methods, we believe that combining the new inference algorithm and M3Ns will achieve a further improvement. While for reranking method, affix features degrade performance a little. The reason is that, in perceptron training stage, candidates are generated by CRFs model learned on 90% training samples rather than the full data set, so the best candidate is less accurate due to small training data size (only about 600 samples), which significantly affect the quality of perceptron learning. SHO-CRFs also exceed SHMM which suffers from numerous local minimum traps that riddle the cost surface.

We also investigate the effect of affix length to see which order is high enough to capture such long distance features. Hence, affixes of length 1,2,..., 8 are gradually added. The results of are shown in Figure 6. The accuracy increases slowly when affix length

[1]www.curlewcommunications.co.uk/wordlist.html

Table 2. Comparison results on OCR data set

Algorithm	Accuracy	Training time(m)
CRFs	80.92%	7.4
M3Ns (linear)	80.5%	-
M3Ns (quad)	86.5%	-
M3Ns (cubic)	87.5%	-
100-Best Rerank	78.9%	66
SHMM	79.05%	26
SHO-CRFs	**88.52**%	29

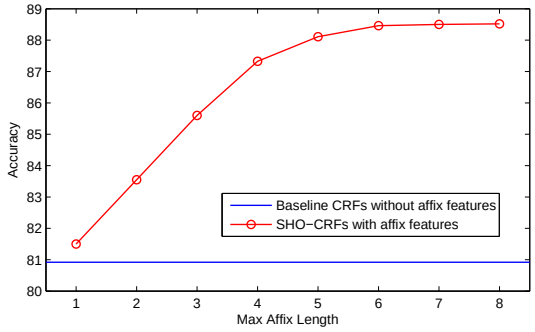

Figure 6. Effect of Affix Length

> 6. So a 5th order CRF is enough to capture affix features, however, such an order is too high for conventional CRFs.

5.2. Chinese Organization Name Recognition

Our second experiment is Chinese Organization (ORG) Name Recognition task, which is the most difficult subtask of Chinese named entity recognition, since the ORG names are often very long and complexly structured. In this task, each Chinese character of sentence is to be assigned with a label $y \in \{B, I, E, S, O\}$, where "B, I, E" means that current character is the beginning, middle, end of a multi-character ORG name respectively, "S" means a single character ORG name, "O" means other. We use MSRA Chinese named entity corpus in SIGHAN 2008 (Jin & Chen, 2008). A baseline CRF model is trained using local features, including first order transition features and surrounding character unigrams and bigrams within a window of 5 characters.

We derived some linguistic characteristics by roughly going through training corpus and represented them by higher order features. Some of them are listed in table 3. Finally, 28 higher order features are defined, a dictionary contains about 20K manually col-

Table 3. Some higher order features in ORG recognition task

Fired condition	Interpretation	Example
$\mathbf{x}_{s:t}$ = location + ...+ suffix AND is labeled as a ORG	Some ORG names have the structure: location + ... + suffix	黄/B 骅/I 市/I 中/I 医/I 精/I 神/I 病/I 专/I 科/I 医/I 院/E (Huanghua Chinese medical psychiatric hospital) Where location = 黄骅市(Huanghua City), suffix = 医院(hospital)
$\mathbf{x}_{s:t}$ =" ..." or (...) or 《...》 AND only one of y_s,y_t is O	Negative feature, to punish those inconsistently labeled symmetric mark pairs	（/O 简/O 称/O 青/B 民/I 盟/E ）/O (（called Qingmin League for short ）) Both brackets should be labeled as O
$\mathbf{x}_{s:t}$ = 、+ $\mathbf{x_1}$ + 、+ ...+ $\mathbf{x_n}$ + 等+ ...+ suffix AND each $\mathbf{x_i}$ is labeled as ORG	Each unit in parallel structures is possibly a ORG.	韩/O 国/O 的/O 现/B 代/E 、/O 乐/B 喜/I 金/I 星/E 、/O 三/B 星/E 等/O 几/O 家/O 大/O 型/O 企/O 业/O 集/O 团/O (Several large-scale enterprise groups such as Hyundai, LG, Samsung) Where $\mathbf{x_1}$ = 现代(Hyundai), $\mathbf{x_2}$ = 乐喜金星(LG), $\mathbf{x_3}$ = 三星(Samsung), suffix = 集团(group)

Table 4. Chinese ORG recognition result

	F1	Training Time(m)	Additional Resources
CRFs	84.02%	42	None
10-Best Rerank	84.81%	375	20K items including location names, company names etc.
20-Best Rerank	84.82%	375	
50-Best Rerank	84.92%	375	
100-Best Rerank	85.03%	375	
SHO-CRFs	**88.59**%	164	
Official best (Yang et al., 2008)	**90.48**%	-	1998 People' s Daily corpus, 40K location names and 300K ORG names
Official 2nd (Yu et al., 2008)	88.40%	-	Additional Chinese word segmenter and POS tagger. A dictionary contains 445K words, 30K location names, 55K ORG names, etc.

lected items including suffixes, location names, company names are used for detecting high order features, i.e., $b(\mathbf{x}, t)$.

For completeness, we compared our SHO-CRFs to the candidate reranking algorithm and the top 2 official runs in SIGHAN 2008 open test[2], results are summarized in table 4. SHO-CRFs significantly outperform N-best reranking with the same feature set and achieve the second best performance using much less resources.

6. Conclusion

We describe SHO-CRFs which could handle local and sparse higher order features together using a novel tractable exact inference. Our work is motivated by the sparseness of long distance features in real applications. Though the computational complexity is not polynomial theoretically, experiments on OCR task and Chinese organization name recognition task clearly demonstrate the efficiency in real practice.

[2]In fact, the top 2 official runs in SIGHAN achieve 99.86% and 99.2% F-score, we did not list them out because they used a larger training corpus that is the super set of testing data.

Acknowledgments

We appreciate for the suggestions by the four reviewers that help improve this paper in many aspects. This work was (partially) funded by Chinese NSF 60673038, Doctoral Fund of Ministry of Education of China 200802460066, Shanghai Science and Technology Development Funds 08511500302, and Shanghai Leading Academic Discipline Project, Project number: B114.

References

Collins, M. (2002a). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *Proceedings of Empirical Methods in Natural Language Processing* (pp. 1–8).

Collins, M. (2002b). Ranking algorithms for named entity extraction: Boosting and the voted perceptron. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 489–496).

Galassi, U., Giordana, A., & Saitta, L. (2007). Struc-

tured hidden markov model: A general framework for modeling complex sequences. *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing* (pp. 290–301).

Jin, G., & Chen, X. (2008). The fourth international chinese language processing bakeoff: Chinese word segmentation, named entity recognition and chinese pos tagging. *Proceedings of Sixth Special Interest Group on Chinese Language Processing Workshop* (pp. 69–81).

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the 18th International Conference on Machine Learning* (pp. 282–289).

Roth, D., & tau Yih, W. (2005). Integer linear programming inference for conditional random fields. *Proceedings of the 22nd International Conference on Machine learning.* (pp. 736–743).

Sarawagi, S., & Cohen, W. (2004). Semi-markov conditional random fields for information extraction. *Advances in Neural Information Processing Systems* (pp. 1185–1192).

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. *Advances in Neural Information Processing Systems* (pp. 25–32).

Yang, F., Zhao, J., & Zou, B. (2008). CRFs-based named entity recognition incorporated with heuristic entity list searching. *Proceedings of Sixth Special Interest Group on Chinese Language Processing Workshop* (pp. 171–174).

Yu, X., Lam, W., Chan, S.-K., Wu, Y., & Chen, B. (2008). Chinese NER using CRFs and logic for the fourth sighan bakeoff. *Proceedings of Sixth Special Interest Group on Chinese Language Processing Workshop* (pp. 102–105).

A. Proof of proposition 2

Suppose $S = \{\mathbf{Z}_{\mathbf{i}s_i:t_i}\}$ are all configurations of $\mathbf{x}$. $\pi_{s:t}$ can be derived as follows: $\pi_{s:t}^{(0)} = \mathbf{U}_{s:t}$; if $s_i \leq t < t_i$, $\pi_{s:t}^{(i)}$ is the refinement of partition $\{E_{s:t}(\mathbf{Z}_{\mathbf{i}s_i:t_i}), \overline{E_{s:t}(\mathbf{Z}_{\mathbf{i}s_i:t_i})}\}$ and $\pi_{s:t}^{(i-1)}$, otherwise, $\pi_{s:t}^{(i)} = \pi_{s:t}^{(i-1)}$. Finally $\pi_{s:t} = \pi_{s:t}^{(|S|)}$. The derivation of $\pi_{r:t+1}$ is similar. We shall prove that for any i, $\mathbf{A}_{s:t}^{(i)} \in \pi_{s:t}^{(i)}$, $\mathbf{B}_{r:t+1}^{(i)} \in \pi_{r:t+1}^{(i)}$, proposition 2 holds.

For $i = 0$, $\pi_{s:t}^{(0)} = \mathbf{U}_{s:t}$, $\pi_{s:t}^{(0)} = \mathbf{U}_{r:t+1}$, let $Y_{t+1} = U$, so proposition 2 holds.

Suppose proposition 2 holds for $i = k$, that is, there exists $Y_{t+1}^{(k)}$ so that $\mathbf{A}_{s:t}^{(k)} \times Y_{t+1}^{(k)} \subseteq \mathbf{U}_{s:r-1} \times \mathbf{B}_{r:t+1}^{(k)}$, $\mathbf{A}_{s:t}^{(k)} \times \overline{Y_{t+1}^{(k)}} \cap \mathbf{U}_{s:r-1} \times \mathbf{B}_{r:t+1}^{(k)} = \emptyset$. Then for $i = k+1$, there are 5 cases:

Case 1, $s_{k+1} = t_{k+1}$. We have $\pi_{s:t}^{(k+1)} = \pi_{s:t}^{(k)}$, $\pi_{r:t+1}^{(k+1)} = \pi_{r:t+1}^{(k)}$, so proposition 2 holds.

Cases 2-5 assume that $s_{k+1} < t_{k+1}$.

Case 2, $s_{k+1} \geq t+2$ or $t_{k+1} \leq t$. Like case 1, neither partition changes, so proposition 2 holds.

Case 3, $s_{k+1} \leq t$, $t_{k+1} \geq t+2$. Both partitions change. $\mathbf{A}_{s:t}^{(k)}$ is split into two parts: $\mathbf{A}_{s:t}^{(k+1)} = \mathbf{A}_{s:t}^{(k)} \cap E_{s:t}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}})$, $\mathbf{C}_{s:t}^{(k+1)} = \mathbf{A}_{s:t}^{(k)} \cap \overline{E_{s:t}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}})}$, $\mathbf{B}_{r:t+1}^{(k)}$ is split into two parts: $\mathbf{B}_{r:t+1}^{(k+1)} = \mathbf{B}_{r:t+1}^{(k)} \cap E_{r:t+1}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}})$, $\mathbf{D}_{r:t+1}^{(k+1)} = \mathbf{B}_{r:t+1}^{(k)} \cap \overline{E_{r:t+1}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}})}$.

For $\mathbf{A}_{s:t}^{(k+1)}$, $\mathbf{B}_{r:t+1}^{(k+1)}$, let $Y_{t+1}^{(k+1)} = Y_{t+1}^{(k)} \cap E_{t+1:t+1}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}})$. We have $\mathbf{A}_{s:t}^{(k+1)} \times Y_{t+1}^{(k+1)} \subseteq \mathbf{A}_{s:t}^{(k+1)} \times Y_{t+1}^{(k)} \subseteq \mathbf{U}_{r:s-1} \times \mathbf{B}_{r:t+1}^{(k)}$, and $\mathbf{A}_{s:t}^{(k+1)} \times Y_{t+1}^{(k+1)} \subseteq E_{s:t}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}}) \times E_{t+1:t+1}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}}) \subseteq \mathbf{U}_{r:s-1} \times E_{r:t+1}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}})$. So we get $\mathbf{A}_{s:t}^{(k+1)} \times Y_{t+1}^{(k+1)} \subseteq \mathbf{U}_{r:s-1} \times \mathbf{B}_{r:t+1}^{(k+1)}$. Notice that $\overline{Y_{t+1}^{(k+1)}} = \overline{Y_{t+1}^{(k)}} \cup \overline{E_{t+1:t+1}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}})}$. Since $\mathbf{A}_{s:t}^{(k)} \times \overline{Y_{t+1}^{(k)}} \cap \mathbf{U}_{s:r-1} \times \mathbf{B}_{r:t+1}^{(k)} = \emptyset$, $\mathbf{A}_{s:t}^{(k+1)} \subseteq \mathbf{A}_{s:t}^{(k)}$,$\mathbf{B}_{r:t+1}^{(k+1)} \subseteq \mathbf{B}_{r:t+1}^{(k)}$, so $\mathbf{A}_{s:t}^{(k+1)} \times \overline{Y_{t+1}^{(k)}} \cap \mathbf{U}_{s:r-1} \times \mathbf{B}_{r:t+1}^{(k+1)} = \emptyset$. Since $\mathbf{U}_{s:t} \times \overline{E_{t+1:t+1}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}})} \cap \mathbf{U}_{s:r-1} \times E_{r:t+1}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}}) = \emptyset$, $\mathbf{A}_{s:t}^{(k+1)} \subseteq \mathbf{U}_{s:t}$, $\mathbf{B}_{r:t+1}^{(k+1)} \subseteq E_{r:t+1}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}})$, so $\mathbf{A}_{s:t}^{(k+1)} \times \overline{E_{t+1:t+1}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}})} \cap \mathbf{U}_{s:r-1} \times \mathbf{B}_{r:t+1}^{(k+1)} = \emptyset$. Hence we get $\mathbf{A}_{s:t}^{(k+1)} \times \overline{Y_{t+1}^{(k+1)}} \cap \mathbf{U}_{s:r-1} \times \mathbf{B}_{r:t+1}^{(k+1)} = \emptyset$. So proposition 2 holds for $\mathbf{A}_{s:t}^{(k+1)}$, $Y_{t+1}^{(k+1)}$, $\mathbf{B}_{r:t+1}^{(k+1)}$. Similarly, we have, proposition 2 holds for $\mathbf{C}_{s:t}^{(k+1)}$, $W_{t+1}^{(k+1)} = \emptyset$, $\mathbf{B}_{r:t+1}^{(k+1)}$; and $\mathbf{A}_{s:t}^{(k+1)}$, $V_{t+1}^{(k+1)} = Y_{t+1}^{(k)} \cap \overline{E_{t+1:t+1}(\mathbf{Z}_{\mathbf{k+1}s_{k+1}:t_{k+1}})}$, $\mathbf{D}_{r:t+1}^{(k+1)}$; and $\mathbf{C}_{s:t}^{(k+1)}$, $Y_{t+1}^{(k)}$, $\mathbf{D}_{r:t+1}^{(k+1)}$, we omit the detail due to space limit.

Case 4, $s_{k+1} = t+1$. Only $\pi_{r:t+1}$ changes, $\mathbf{B}_{r:t+1}^{(k)}$ is split into two parts, proof is similar with case 3.

Case 5, $t_{k+1} = t+1$. Only $\pi_{s:t}$ changes, proof is similar with case 3. □