Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Sequence Labeling with Non-Negative Weighted Higher Order Features

Xian Qian **Yang Liu**
The University of Texas at Dallas
qx,yangl@hlt.utdallas.edu

Abstract

In sequence labeling, using higher order features leads to high inference complexity. A lot of studies have been conducted to address this problem. In this paper, we propose a new exact decoding algorithm under the assumption that weights of all higher order features are non-negative. In the worst case, the time complexity of our algorithm is quadratic on the number of higher order features. Comparing with existing algorithms, our method is more efficient and easier to implement. We evaluate our method on two sequence labeling tasks: Optical Character Recognition and Chinese part-of-speech tagging. Our experimental results demonstrate that adding higher order features significantly improves the performance without much additional inference time.

Introduction

In a sequence labeling task, we are given an input sequence of length l, $\mathbf{x} = x_1 \dots x_l$, and need to assign each component x_t with a label to generate label sequence $\mathbf{y} = y_1 \dots y_l$. The sequence labeling problems arise in a variety of applications, such as assigning part-of-speech (POS) tags to words in POS tagging tasks, labeling words in sentences with their entity types and boundaries in named entity recognition problems, recognizing handwritten character images in Optical Character Recognition (OCR) tasks. In many sequence labeling problems, there is important structural information among the elements in the label sequence y. Structured learning models have been successfully applied in these problems because of their ability of using arbitrary structured information, especially for discriminative models that benefit from a large amount of even redundant features, such as Conditional Random Fields (CRFs) (Lafferty, McCallum, and Pereira 2001), Averaged Perceptron (Collins 2002a), Max Margin Markov Networks (Taskar, Guestrin, and Koller 2003), online Passive Aggressive Learning (Crammer et al. 2006). In these models, a feature or model of order k encodes the dependency between $\mathbf{x}$ and $(k+1)$ consecutive elements in $\mathbf{y}$. Using higher order features in sequence labeling can potentially lead to an exponential computational complexity in inference. Hence, structured information is usually assumed to exist only between adjacent components of $\mathbf{y}$, resulting in first order Markov models that are much less expressive and have been widely adopted for various sequence labeling tasks.

In this paper, we propose a new decoding algorithm under the following assumption: weights of higher order features are non-negative. Such an assumption is reasonable in cases where higher order features are derived by human prior knowledge (since we usually expect these features show positive effect on predictions), or the frequent transition features are extracted from training data. For example, in the named entity recognition task, one higher order feature is true if the word sequence between "professor" and "said" is labeled as a person name. We expect that adding this feature will generate more person names, thus its weight should be non-negative. Note that this is not always true. Take the following sentence as an example: *the professor and the student said*. In this case, we expect that other local features (e.g., word 'and' is unlikely to be part of a person name) are dominant and can help make a correct decision.

The basic idea behind our algorithm is simple: given a set S and its maximal element e^*, if we increase the value of an element $e' \in S$, then the updated maximum is either e^* or e', and there is no need to consider the other elements. Therefore, if all higher-order features are non-negative weighted, then we can derive the best label sequence y^* in the following way. For each component x_t, we first neglect the higher order features and use standard Viterbi algorithm to obtain a suboptimal label sequence $y'_1 y'_2 \dots y'_t$; then we update the scores of some label sequences that have higher order features. The optimal label sequence is among $y'_1 \dots y'_t$ and the ones with updated scores.

We conduct experiments on OCR and Chinese POS tagging tasks. For the OCR task, we demonstrate that frequent higher order transition features can significantly improve the performance while requiring only 30% additional time. For the POS tagging task, we manually designed a small number of higher order features based on observations of the training corpus. Our experiments show that adding these features results in 5.3% relative error reduction at the cost of about 10% additional time. Our result is the best reported performance for this task.

Related Work

Existing inference algorithms for higher order models can be divided into five categories: approximate inference, can-

didate reranking, semi-Markov model, Lagrange relaxation, and sparse higher order models.

The approximate algorithms such as Loopy Belief Propagation (LBP), Gibbs sampling (Finkel, Grenager, and Manning 2005), Integer Linear Programming (Roth and Yih 2005) are efficient in practice; however, they are not guaranteed to converge to a good approximation, or they may not even converge.

Reranking methods (Collins 2002b; Dinarelli, Moschitti, and Riccardi 2012) adopt a two-step framework where higher order features are incorporated in the second step to rerank candidate predictions generated by local models in the first step. This method is computationally efficient; however the effectiveness of the reranking module is restricted by the performance of the local models and the number of candidates.

For Semi-Markov models such as semi-CRF (Sarawagi and Cohen 2004), the inference is exact and in polynomial time; however, the high order features in these models have to be in the form of 'segments' and other types can not be handled.

Dual decomposition (Rush et al. 2010) is a special case of Lagrangian relaxation. It relies on standard decoding algorithms as oracle solvers for sub-problems, together with a simple method for forcing agreement between the different oracles. It has been successfully applied in several parsing and joint learning tasks, however, it is a general method and inefficient for special cases.

Similar to dual decomposition, cascaded inference (Weiss, Sapp, and Taskar 2010) also ensembles tractable sub-models as part of a structured prediction cascade. It does not enforce agreement between sub-models, but rather filter the space of possible outputs by simply adding and thresholding the max-marginals of each constituent model.

Recently, sparse higher order models are proposed, such as Sparse Higher Order CRFs (SHO-CRFs) (Qian et al. 2009). They learn and predict with high order features efficiently under the pattern sparsity assumption. Though the complexity is not polynomial, SHO-CRFs are still efficient in practice because of the feature sparsity. Furthermore, the inference has been demonstrated to be polynomial when all the features depend on consecutive label sequences (Ye et al. 2009). The main drawback of this method is that it needs to induce partitions of label sequences before inference. Such induction is even more time consuming than inference, and is difficult to implement.

Unlike SHO-CRFs, our algorithm does not require the state partition information, rather it only needs the relations in the label sequences that higher order features depend on. In addition, our method is much easier to implement.

Problem Definition

Given a sequence $\mathbf{x} = x_1, \dots x_l$, the structured linear classifier assigns a score to each label sequence $\mathbf{y} = y_1, \dots, y_l$, defined as:

$$score(\mathbf{x}, \mathbf{y}) = \sum_t \sum_i w_i f_i(\mathbf{x}, \mathbf{y}_{s:t})$$

where $\mathbf{y}_{s:t}$ denotes the label sequence $y_s, y_{s+1}, \dots, y_t$, and $f_i(\mathbf{x}, \mathbf{y}_{s:t})$ is the feature function that encodes the dependency between $\mathbf{x}$ and $\mathbf{y}_{s:t}$, and w_i is the weight of feature f_i.

In this paper, for simplicity, we will focus on the case of binary features. However, our results extend easily to the general real valued cases. Generally, the feature function has the following form:

$$f_i(\mathbf{x}, \mathbf{y}_{s:t}) = \begin{cases} b(\mathbf{x}, t) I(\mathbf{y}_{s:t}, \mathbf{z}) & \text{if } \mathbf{y}_{s:t} = \mathbf{z} \\ 0 & \text{otherwise} \end{cases}$$

where both $b(\cdot)$ and $I(\cdot)$ are indicator functions. $b(\mathbf{x}, t)$ can be used to represent information such as whether x_t is capitalized or matches a particular word in language processing tasks. $\mathbf{z}$ is a label sequence specified by feature f_i, for example, [*verb*, *noun*] in POS tagging, capturing state transition *verb* $\rightarrow$ *noun*. $I(\mathbf{y}_{s:t}, \mathbf{z})$ indicates whether $\mathbf{y}_{s:t} = \mathbf{z}$. We use the same terminology as in (Ye et al. 2009), calling the sequence of $\mathbf{z}$ the label **pattern** of f_i.

The product of $w_i f_i(\mathbf{x}, \mathbf{z}_{s:t})$ is the **score** of pattern $\mathbf{z}_{s:t}$,[1] denoted as $\phi(\mathbf{x}, \mathbf{z}_{s:t})$. For binary features, the score is simply w_i. The order of the feature/pattern is the length of its label pattern minus one, i.e., $t - s$. For example, in first order Markov model, the transition feature $f(\mathbf{x}, y_{t-1:t})$ is an order-1 feature. We call a feature/pattern low order feature/pattern if its order is ≤ 1, or higher order feature/pattern, if its order is > 1.

We use averaged perceptron for model training (i.e., weight estimation) in this study. In decoding, the input is the observation $\mathbf{x}$, feature functions $\{f_i\}$ and their weights $\{w_i\}$, and the output is the label sequence with the highest score: $\mathbf{y}^* = \arg\max_{\mathbf{y}} score(\mathbf{x}, \mathbf{y})$. The problem is equivalent to searching for $\mathbf{y}^*$ given the scores of all patterns.

Without loss of generality, in the remainder, we assume that all the features are non-negative. If not (i.e., $f_i < 0$), we can derive an equivalent model by replacing f_i with $f'_i = -f_i$ and $w'_i = -w_i$. Then under the assumption of non-negative features, the non-negative weight assumption is equivalent to $\phi(\mathbf{x}, \mathbf{z}^j) \geq 0$ (ϕ is the product of weight and feature), for any higher order pattern $\mathbf{z}^j$.

Decoding Algorithm

We use the following additional notations in our decoding algorithm.

$\mathbf{y}[\mathbf{z}_{s:t}]$ denotes a label sequence $\mathbf{y}_{1:t}$ ending with $\mathbf{z}_{s:t}$, i.e., $\mathbf{y}_{s:t} = \mathbf{z}_{s:t}$. Specially, for the optimal sequence y^*,

$$\mathbf{y}^*[\mathbf{z}_{s:t}] = \arg\max_{\mathbf{y}_{1:t}} score(\mathbf{x}, \mathbf{y}[\mathbf{z}_{s:t}])$$

Let $\mathbf{y}1 \oplus \mathbf{y}2$ denote the concatenation of two sequences. The cost function $cost(\mathbf{x}, \mathbf{y}_{s:t})$ is the sum of the weights of state and transition features within $\mathbf{y}_{s:t}$ (as used in first-order Markov models):

$$cost(\mathbf{x}, \mathbf{y}_{s:t}) = \sum_{r=s+1}^{t} \left(\sum_i w_i f_i(\mathbf{x}, y_r) + \sum_j w_j f_j(\mathbf{x}, \mathbf{y}_{r-1:r}) \right)$$

For a pattern z^j, its score $\phi(\mathbf{x}, \mathbf{z}^j)$ is simplified as ϕ_j.

[1] When there is no ambiguity, we use $f_i(\mathbf{x}, \mathbf{z}_{s:t})$ to represent feature $f_i(\mathbf{x}, \mathbf{y}_{s:t})$ when the subsequence of y matches z: $\mathbf{y}_{s:t} = \mathbf{z}$.

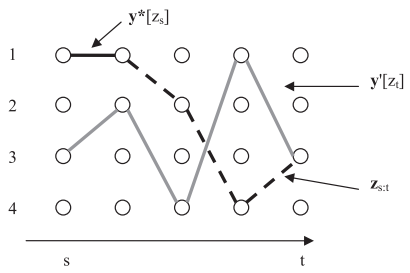

Figure 1: Algorithm for one higher order pattern $\mathbf{z}_{s:t} = 1243$. Gray line denotes the suboptimal $\mathbf{y}'[z_t]$ derived by Viterbi algorithm, solid line denotes the optimal $\mathbf{y}^*[z_s]$, dashed line denotes $\mathbf{z}_{s:t}$. Optimal $\mathbf{y}^*[z_t]$ is either the gray line or the concatenation of black and dashed lines.

One Higher Order Pattern Case

In this section, we study the simplest case, where only one higher order pattern $\mathbf{z}_{s:t}$ exists. The algorithm starts at the first node x_0, and finds the optimal label sequence $\mathbf{y}^*[y_r]$ for each $r < t$ and $y_r \in \mathcal{Y}$ using the standard Viterbi algorithm. For the t^{th} node with label z_t, we first obtain the suboptimal label sequence $\mathbf{y}'[z_t]$ using low order features by enumerating y_{t-1}:

$$y'_{t-1} = \arg\max_{y_{t-1}} \{score(\mathbf{x}, \mathbf{y}^*[y_{t-1}]) + cost(\mathbf{x}, y_{t-1} z_t)\}$$
$$\mathbf{y}'[z_t] = \mathbf{y}^*[y'_{t-1}] \oplus z_t$$

Then the optimal $\mathbf{y}^*[z_t]$ is either the original $\mathbf{y}'[z_t]$ or $\mathbf{y}^*[z_s] \oplus \mathbf{z}_{s+1:t}$:

$$\mathbf{y}^*[z_t] = \begin{cases} \mathbf{y}^*[z_s] \oplus \mathbf{z}_{s+1:t} & \text{if } score(\mathbf{x}, \mathbf{y}'[z_t]) \le score(\mathbf{x}, \mathbf{y}^*[z_s]) \\ & \quad + cost(\mathbf{x}, \mathbf{z}_{s:t}) + \phi(\mathbf{x}, \mathbf{z}_{s:t}) \\ \mathbf{y}'[z_t] & \text{otherwise} \end{cases}$$

This equation still holds if the suboptimal $\mathbf{y}'[z_t]$ and $\mathbf{y}^*[z_s] \oplus \mathbf{z}_{s+1:t}$ are the same. In such cases, we only need to update the score of $\mathbf{y}'[z_t]$.

The algorithm is illustrated in Figure 1.

Two Pattern Case

Suppose there are two higher order patterns $\mathbf{z}^1_{s_1:t_1}, \mathbf{z}^2_{s_2:t_2}$. Without loss of generality, we assume $t_1 \le t_2$, and $s_1 \le s_2$ if $t_1 = t_2$. There are four relations between them: separated, disjoint, subsequence, pre-sequence.

$\mathbf{z}^1, \mathbf{z}^2$ are **separated**, if $t_1 \le s_2$. We could use the algorithm for one pattern case twice: get $\mathbf{y}^*[z^1_{t_1}]$ when neglecting $\mathbf{z}^2$, and get $\mathbf{y}^*[z^2_{t_2}]$ when neglecting $\mathbf{z}^1$. See Figure 2.

$\mathbf{z}^1, \mathbf{z}^2$ are **disjoint**, if $t_1 > s_2$ and $\mathbf{z}^1_{s_2:t_1} \ne \mathbf{z}^2_{s_2:t_1}$. For example, in POS tagging task, two transition patterns *verb → conjunction → verb* and *noun → conjunction → noun* for a word trigram are disjoint. The algorithm is the same as the separated case, since there is no label sequence $\mathbf{y}_{1:l}$ satisfying $\mathbf{y}_{s_1:t_1} = \mathbf{z}^1_{s_1:t_1}$ and $\mathbf{y}_{s_2:t_2} = \mathbf{z}^2_{s_2:t_2}$, which means that the scores of all $\mathbf{y}[\mathbf{z}^2_{s_2:t_2}]$ (including the optimal y^*) are not affected by ϕ_1. See Figure 3.

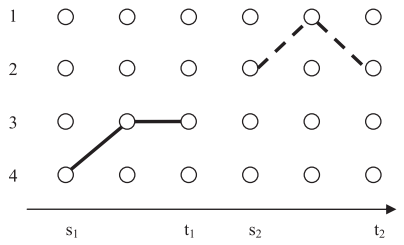

Figure 2: Separated case: $\mathbf{z}^1_{s_1:t_1}$ (solid) and $\mathbf{z}^2_{s_2:t_2}$ (dashed) are separated. Get $\mathbf{y}^*[z_{t_1} = 3]$ and $\mathbf{y}^*[z_{t_2} = 2]$ independently.

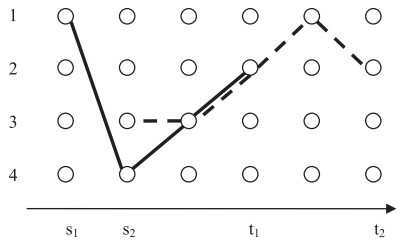

Figure 3: Disjoint case: $\mathbf{z}^1_{s_1:t_1}$ (solid) and $\mathbf{z}^2_{s_2:t_2}$ (dashed) are disjoint. Get $\mathbf{y}^*[z_{t_1} = 2]$ and $\mathbf{y}^*[z_{t_2} = 2]$ independently.

$\mathbf{z}^1$ is **subsequence** of $\mathbf{z}^2$ if $s_1 \ge s_2$ and $\mathbf{z}^1_{s_1:t_1} = \mathbf{z}^2_{s_1:t_1}$. In other words, all label sequences $\mathbf{y}[\mathbf{z}^2_{s_2:t_2}]$ satisfy $\mathbf{y}_{s_1:t_1} = \mathbf{z}^1_{s_1:t_1}$, hence we could use the same algorithm as the separated case by letting $\phi_2 = \phi_2 + \phi_1$. See Figure 4.

$\mathbf{z}^1$ is **pre-sequence** of $\mathbf{z}^2$ if $s_1 < s_2 < t_1 < t_2$ and $\mathbf{z}^1_{s_2:t_1} = \mathbf{z}^2_{s_2:t_1}$. For example, in POS tagging task, for words *have a red book*, transition pattern *verb → article → adjective* is the pre-sequence of the pattern *article→ adjective → noun* . Note that if $t_1 = t_2$, then $\mathbf{z}^2$ is the subsequence of $\mathbf{z}^1$.

For the pre-sequence case, the algorithm is similar to the separated case, except that now we need to consider the combination of patterns when searching for $\mathbf{y}^*[\mathbf{z}^2_{s_2:t_2}]$.

First we get the best $\mathbf{y}^*[\mathbf{z}^2_{s_2:t_1}]$:

$$\mathbf{y}^*[\mathbf{z}^2_{s_2:t_1}] = \begin{cases} \mathbf{y}^*[z^1_{s_1}] \oplus \mathbf{z}^1_{s_1+1:t_1} & \text{if } score(\mathbf{x}, y^*[z^1_{s_1}]) + \phi_1 + cost(\mathbf{x}, \mathbf{z}^1) \\ & \quad > score(\mathbf{x}, \mathbf{y}^*[z^2_{s_2}]) + cost(\mathbf{x}, \mathbf{z}^2_{s_2:t_1}) \\ \mathbf{y}^*[z^2_{s_2}] \oplus \mathbf{z}^2_{s_2+1:t_1} & \text{otherwise} \end{cases}$$

Then the optimal $\mathbf{y}^*[z^2_{t_2}]$ is

$$\mathbf{y}^*[z^2_{t_2}] = \begin{cases} \mathbf{y}^*[z^2_{s_2:t_1}] \oplus \mathbf{z}^2_{t_1+1:t_2} & \text{if } score(\mathbf{x}, \mathbf{y}^*[\mathbf{z}^2_{s_2:t_1}]) + \phi_2 \\ & \quad + cost(\mathbf{x}, \mathbf{z}^2_{t_1:t_2}) > score(\mathbf{x}, \mathbf{y}'[z^2_{t_2}]) \\ \mathbf{y}'[z^2_{t_2}] & \text{otherwise} \end{cases}$$

where $\mathbf{y}'[z^2_{t_2}]$ is the suboptimal label sequence obtained by low order patterns. See Figure 5.

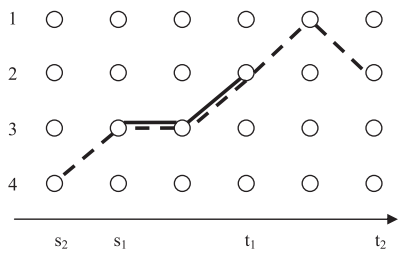

Figure 4: Subsequence case: $\mathbf{z}^1_{s_1:t_1}$ (solid) is subsequence of $\mathbf{z}^2_{s_2:t_2}$ (dashed). Let $\phi_2 = \phi_2 + \phi_1$, then get $\mathbf{y}^*[z_{t_1} = 2]$ and $\mathbf{y}^*[z_{t_2} = 2]$ independently.

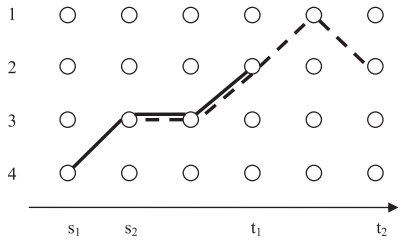

Figure 5: Pre-sequence case: $\mathbf{z}^1_{s_1:t_1}$ (solid) is pre-sequence of $\mathbf{z}^2_{s_2:t_2}$ (dashed). We first get $\mathbf{y}^*[z^2_{s_2:t_1}]$ (ending with the common sequence of the solid and dashed), then get $\mathbf{y}^*[z^2_{t_2}]$.

General Case

As discussed in the previous section, the only case that we can not treat the patterns independently is the pre-sequence case. Therefore, in the general case, for each higher order pattern $\mathbf{z}_{s:t}$, we need to consider all of its pre-sequences before getting $\mathbf{y}^*[z_t]$. Formally, suppose $P = \{\mathbf{z}^i_{s_i:t_i}\}$ are the pre-sequences of $\mathbf{z}_{s:t}$, then $\mathbf{y}^*[\mathbf{z}_{s:t}]$ can be derived recursively. For $s < r \leq t$, $\mathbf{y}^*[\mathbf{z}_{s:r}]$ is from one of the two cases: (i) $\mathbf{y}^*[\mathbf{z}_{s:r-1}] \oplus z_r$ whose score is $score(\mathbf{x}, \mathbf{y}^*[\mathbf{z}_{s:r-1}]) + cost(\mathbf{x}, \mathbf{z}_{r-1:r})$; (ii) $\mathbf{y}^*[z^k_{s_k}] \oplus \mathbf{z}^k_{s_k+1:r}$, where

$$k = \arg\max_{j, \mathbf{z}^j \in P, t_j = r} score(\mathbf{x}, \mathbf{y}^*[\mathbf{z}^j])$$

$$score(\mathbf{x}, \mathbf{y}^*[\mathbf{z}^j]) = score(\mathbf{x}, \mathbf{y}^*[\mathbf{z}^j_{s_j:r-1}]) + \phi_j + cost(\mathbf{x}, \mathbf{z}^j_{r-1:r}) \quad (1)$$

The label sequence with the highest score is selected. An example is illustrated in Figure 6.

Finally we summarize the decoding algorithm in Algorithm 1. Steps 1-7 find the pre-sequences and subsequences of each higher order pattern. Step 8 adds the subsequence scores to the corresponding patterns. Then for each $t, y_t \in \mathcal{Y}$, steps 11 and 12 search the suboptimal label sequence $\mathbf{y}'[y_t]$ using low order patterns. Steps 13-18 update $\mathbf{y}'[y_t]$ by considering all higher order patterns ending with y_t. The optimal $\mathbf{y}^*[y_t]$ is obtained in step 19. Steps 20-27 search $\mathbf{y}^*[\mathbf{z}^i_{s_i:t}]$, the best label sequence ending with the pre-sequence of higher order pattern $\mathbf{z}^i$ satisfying $z^i_t = y_t$.

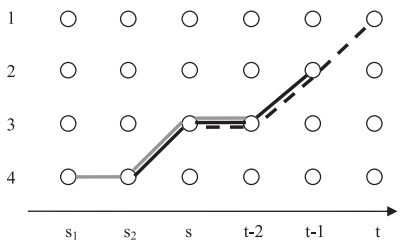

Figure 6: General case: $\mathbf{z}^1_{s_1:t-2}$ (gray) and $\mathbf{z}^2_{s_2:t-1}$ (solid) are pre-sequences of $\mathbf{z}_{s:t}$ (dashed), meanwhile $\mathbf{z}^1$ is the pre-sequence of $\mathbf{z}^2$. The best label sequence ending with the dashed line is the extension of the best one ending with the dashed segment, from s to $t-1$, $\mathbf{y}^*[\mathbf{z}_{s:t-1}]$. $\mathbf{y}^*[\mathbf{z}_{s:t-1}]$ is one of the two sequences: the extension of the best one with the first dashed segment $\mathbf{y}^*[\mathbf{z}_{s:t-2}]$; the other is the best one ending with solid line $\mathbf{y}^*[\mathbf{z}^2]$ which is obtained the same way as the dashed line.

Complexity Analysis

We use the following notations:

- l: sequence length.
- $|\mathbf{z}|$: length of pattern $\mathbf{z}$.
- L: max length of pattern $\mathbf{z}$.
- n: number of higher order patterns
- $|\mathcal{Y}|$: size of label set

Time complexity for finding the pre-sequences and subsequences of each pattern is bounded by $L \cdot n^2$ (there are at most L comparisons for each pattern pair). There are n patterns, and for each $\mathbf{z}_{s:t}$, there are at most $n-1$ pre-sequences and L sequences to compare (steps 25,26). The complexity for Viterbi decoding is $l|\mathcal{Y}|^2$ (steps 11-12). Therefore the total complexity of our algorithm is bounded by $(n+L)n + Ln^2 + l|\mathcal{Y}|^2$, which is square in n.

Experiments

We conduct experiments on the Optical Character Recognition (OCR) task and Chinese POS tagging task. We use averaged perceptron in training, and the iteration number is 50 for OCR and 10 for POS tagging. To guarantee that all higher order features $\{f_i\}$ are positively weighted, we simply project the their weights onto non-negative space.

Optical Character Recognition

We use the same data and settings as in (Taskar, Guestrin, and Koller 2003), where 6100 handwritten words with an average length of around 8 characters were divided into 10 folds, each with 600 training and 5500 testing examples. The words were segmented into characters, with the first capital character removed. Each character was depicted by an image of 16×8 binary pixels. In this labeling problem, each x_i is the image of a character, and each y_i is a lower-case letter. The low order features we used are pixel features (including single pixels and combination of 2 adjacent pixels)

Algorithm 1 Decoding Algorithm with Non-Negative Weighted Higher Order Features

Input: Sentence $\mathbf{x} = x_1, \ldots, x_l$, low order patterns and their scores, i.e., $cost(\mathbf{x}, y_{t-1}y_t)$, higher order patterns $\{\mathbf{z}^i_{s_i:t_i} | t_i \leq t_{i+1}\}_{i=1}^n$ and their scores $\{\phi_i\}_{i=1}^n$

Output: Label sequence $\mathbf{y}^*$ with the highest score

Initialize the pre-sequences and subsequences for higher order patterns $P_i = \emptyset$, $S_i = \emptyset$, $\quad 1 \leq i \leq n$
for $1 \leq i < j \leq n$ **do** $\quad \triangleleft$ find pre-sequences and subsequences for each higher order pattern
  if $s_i < s_j < t_i < t_j$ AND $\mathbf{z}^i_{s_j:t_i} = \mathbf{z}^j_{s_j:t_i}$ **do**
    $P_j = P_j \cup \{\mathbf{z}^i\}$ $\quad \triangleleft$ pattern $\mathbf{z}^i$ is the pre-sequence of $\mathbf{z}^j$
  else if $s_j \leq s_i \leq t_i \leq t_j$ AND $\mathbf{z}^i_{s_i:t_i} = \mathbf{z}^j_{s_i:t_i}$ **do**
    $S_j = S_j \cup \{\mathbf{z}^i\}$ $\quad \triangleleft$ pattern $\mathbf{z}^i$ is the subsequences of $\mathbf{z}^j$
  end if
end for
$\phi_i = \phi_i + \sum_{\mathbf{z}^j \in S_i} \phi_j, \quad 1 \leq i \leq n$ $\quad \triangleleft$ Add the scores of subsequence patterns.
for $t = 1, \ldots, l$ **do**
  for $y_t \in \mathcal{Y}$ **do**
    $y'_{t-1} = \arg\max_{y_{t-1}} \{score(\mathbf{x}, \mathbf{y}^*[y_{t-1}]) + cost(\mathbf{x}, y_{t-1}y_t)\}$
    $\mathbf{y}'[y_t] = \mathbf{y}^*[y'_{t-1}] \oplus y_t$
    foreach $\mathbf{z}^i_{s_i:t_i}$ satisfying $t_i = t$ and $z^i_t = y_t$ **do**
      if $score(\mathbf{x}, \mathbf{y}'[y_t]) < score(\mathbf{x}, \mathbf{y}^*[\mathbf{z}^i_{s_i:t-1}]) + cost(\mathbf{x}, \mathbf{z}^i_{t-1:t}) + \phi_i$ **do**
        $score(\mathbf{x}, \mathbf{y}'[y_t]) = score(\mathbf{x}, \mathbf{y}^*[\mathbf{z}^i_{s_i:t-1}]) + cost(\mathbf{x}, \mathbf{z}^i_{t-1:t}) + \phi_i$
        $\mathbf{y}'[y_t] = \mathbf{y}^*[\mathbf{z}^i_{s_i:t-1}] \oplus y_t$
      end if
    end foreach
    $\mathbf{y}^*[y_t] = \mathbf{y}'[y_t]$ $\quad \triangleleft$ get the optimal sequence
    foreach $\mathbf{z}^i_{s_i:t_i}$ satisfying $s_i < t < t_i$ and $z^i_t = y_t$ **do**
      Let $Q = \{\mathbf{z}^j_{s_j:t_j} | \mathbf{z}^j \in P_i, t_j = t\}$
      foreach $\mathbf{z}^j \in Q$
        $score(\mathbf{x}, \mathbf{y}^*[\mathbf{z}^j_{s_j:t}]) = score(\mathbf{x}, \mathbf{y}^*[\mathbf{z}^j_{s_j:t-1}]) + \phi_j + cost(\mathbf{x}, \mathbf{z}^i_{t-1:t})$
      end foreach
      $k = \arg\max_j \{score(\mathbf{x}, \mathbf{y}^*[\mathbf{z}^j_{s_j:t}])\}_{\mathbf{z}^j \in Q}$
      $\mathbf{y}^*[\mathbf{z}^i_{s_i:t}] = \begin{cases} \mathbf{y}^*[z^k_{s_k}] \oplus \mathbf{z}^k_{s_k-1,t} & \text{if } score(\mathbf{x}, \mathbf{y}^*[\mathbf{z}^k_{s_k:t}]) > score(\mathbf{x}, \mathbf{y}^*[z^i_{s_i:t-1}]) + cost(\mathbf{x}, \mathbf{z}^i_{t-1:t}) \\ \mathbf{y}^*[z^i_{s_i}] \oplus \mathbf{z}^i_{s_i-1,t} & \text{otherwise} \end{cases}$
    end foreach
  end for
end for

system	Order	Accuracy	Sec./iter	#iter
M^{3}N (Cubic Kernel)	1	87.5%	-	-
(Ye et al. 2009)	5	85.5%	85*	40
(Qian et al. 2009)	7	**88.52%**	5.8*	300
Dual Decomposition	7	86.75%	1.04	50
Ours	7	**88.53%**	**0.390**	50
Ours (baseline)	1	82.42%	0.319	50

Table 1: Results for OCR task. *Time is not directly comparable to this paper due to different hardware platforms.

and first order transition features. For higher order features, we simply use all the transition features that appeared in the training data with frequency > 10 and order < 8. To study the effect of feature orders, we gradually add features of order $1, 2, \ldots, 7$.

For performance evaluation, we use the averaged character accuracy and running time over the 10 folds. Table 1 shows the results. For our system, we also report the baseline results (without incorporating high order features). For comparison, performance of several state-of-the-art systems is also listed. M^{3}N denotes Max Margin Markov Networks with cubic kernel (Taskar, Guestrin, and Koller 2003). For the dual decomposition algorithm, we treat each higher order pattern as a slave, and the linear chain model with local features as a slave. Viterbi algorithm is used for searching iteratively until all slaves agree. We found that, dual decomposition hardly converges due to lots of slaves, hence we set the maximum iteration number with 10 for efficiency. We can see that in our system, adding higher order features significantly improves the performance, and requires only about 30% additional time. Compared with the best reported system (Qian et al. 2009), our system is competitive and much more efficient.

Figure 7 and 8 show the results when changing the maximum order of features. The gain from adding high order features is more noticable at the beginning, and become less after order 5. We can see that our algorithm is scalable for higher order features, as the running time appears to grow no more than linearly with the maximum order of the features.

Chinese POS Tagging

For POS tagging, to compare with other systems, we use the same split as in SIGHAN2008 using the Chinese Tree Bank

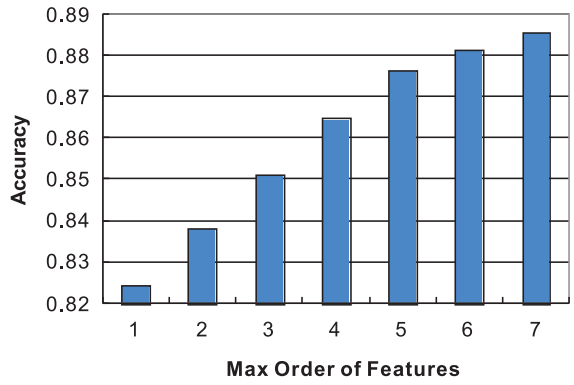

Figure 7: Accuracy on OCR task using different orders of features.

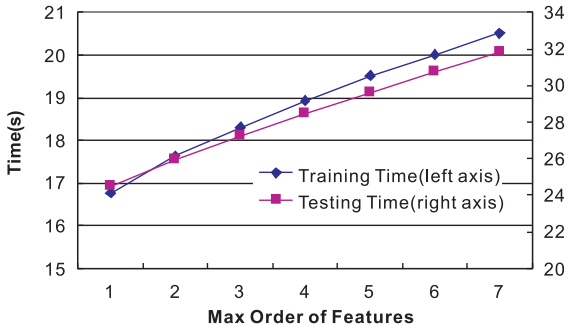

Figure 8: Training/testing time on OCR task using different orders of features.

(CTB) corpus (Jin and Chen 2008). Data statistics are show in Table 2.

The low order features are: (i) word level features, including surrounding word unigrams, bigrams, and word length; (ii) character level features, such as the first and last characters of words; (iii) transition features.

The higher order features we used are manually designed. We derived some linguistic characteristics by examining the training corpus and represented them as binary higher order features. Some example are listed in Figure 9.

Table 3 shows the POS tagging results using our method, as well as from two best systems in the official evaluation using this same corpus. We can see that using the higher order feature yields 5.3% relative error reduction at the cost of only about 10% additional training/testing. The improvement from using higher order features over the low order model is statistically signficant (based on McNemar's test; $p < 0.01$). Our result is the best for this task. Note that the best official system (Wu et al. 2008) benefits from model combination – they combined the results of CRFs (94.0%) and TBL (92.7%) for better performance, while our system is a single model.

system	Accuracy	Train Sec.	Test Sec.
Official Best	94.28	-	-
Official Second	94.01	-	-
Ours (higher order)	**94.41**	**7431**	**26.2**
Ours (low order)	94.04	6919	23.4

Table 3: Results for Chinese POS tagging.

Discussion about Negative Weighted Features

Our algorithm could not be applied to cases where negative weighted features exist. One example is shown in Figure 10, where there are two negative patterns: $\mathbf{z}^1_{0,2}$ and $\mathbf{z}^2_{1,3}$. Suppose at $t = 2$, we get $\mathbf{y}^*[z^1_2] = \mathbf{y}^*[z^2_{1:2}]$ (i.e., sequence ending with $z^2_{1:2}$), and at $t = 3$, we find that the suboptimal $\mathbf{y}'[z^2_3]$ derived by low order patterns ended with $\mathbf{z}^2$. Since $\phi_2 < 0$, we need to check if $\mathbf{y}'[z^2_3]$ is better than the second best $\mathbf{y}[z^2_3]$. It is possible that the second best sequence we identified is: $\mathbf{y}[z^2_3] = \mathbf{y}^*[\mathbf{z}^1 \oplus z^2_3]$. In this case, since $\phi_1 < 0$, the optimal $\mathbf{y}^*[z^2_3]$ may be neither $\mathbf{y}^*[\mathbf{z}^1 \oplus z^2_3]$ nor $\mathbf{y}'[z^2_3]$. We should search for the best $\mathbf{y}[z^2_3]$, satisfying $\mathbf{y}_{0:2} \neq z^1$ and $\mathbf{y}_{1:3} \neq z^2$ to see which one is the optimal. This is equivalent to dividing $\{\mathbf{y}_{0:3}\}$ into 3 partitions, and search for the best $\mathbf{y}[z^2_3]$ from each partition. Hence, state partition is necessary for negative patterns, which is time-consuming as studied by previous work (Qian et al. 2009; Ye et al. 2009).

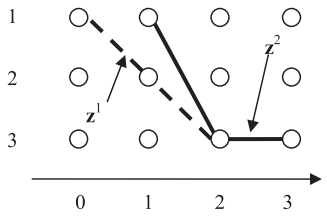

Figure 10: Our algorithm is not designed for cases where negative patterns exist: $\mathbf{z}^1_{0,2}$ and $\mathbf{z}^2_{1,3}$ are negative patterns.

Conclusion

In sequence labeling, incorporating higher order structural information often helps improve labeling performance, however, it is computationally expensive. In this paper, we present a novel exact decoding algorithm for sequence labeling with non-negative weighted higher order features. This assumption often holds in real applications. We demonstrate the efficiency of the algorithm both theoretically and empirically. Experiments on the OCR and Chinese POS tagging tasks show that, the time cost of our algorithm is competitive comparing with the standard Viterbi algorithm, while significantly improving the performance.

	Token	WT	TT	ATN	OOV	$\mathcal{R}_{OOV}$	MT$_{IV}$	$\mathcal{R}_{MT\,IV}$
Training	642246	42133	37	1.1690			334317	0.5205
Testing	59955	9797	35	1.1227	3794	0.0633	30513	0.5089

Table 2: Chinese POS tagging data statistics. WT: number of word type; TT: number of tag type. ATN: Average Tag Number per word. MT$_{IV}$: number of In-Vocabulary (IV) Multi-Tag word. $\mathcal{R}_{MT\,IV}$: coverage rate of IV Multi-Tag words. $\mathcal{R}_{OOV}$: Out of Vocabulary (OOV) word rate.

Definition	Explanation
$\mathbf{x}_{s:s+4} = w_s$、w_{s+2}、w_{s+4} AND $\mathbf{z}_{s:s+4} = y, PU, y, PU, y$, where PU is abbreviation of POS tag 'punctuation', y denotes any POS tag	Coordinating words are likely to have the same POS.
w_s =百分之..., w_{s+1} =至AND $\mathbf{z}_{s:s+2} = CD, CC, CD$, where CD,CC are abbreviations of POS tag 'Cardinal Number' and 'Coordinating Conjunction'.	Phrase like $\mathbf{x}_{s:s+2}$ ='a% to b%' is likely to be labeled with CD, CC, CD.
$\mathbf{x}_{s:s+4}$ =从w_{s+1}到w_{s+4} AND $\mathbf{z}_{s:s+4} = P, y, P, y$, where P is abbreviation of POS tag 'Preposition' y denotes any POS tag	Phrase 'from A to B' is likely to be labeled with P, y, P, y.

Figure 9: Examples of higher order features for POS tagging.

Our algorithm could be directly applied to tree structures or semi-Markov models. However, it is invalid for general structures with circles. One may combine it with other approximate techniques like tree-reweighted message-passing techniques.

Acknowledgments

We thank the three anonymous reviewers for valuable suggestions. This work is supported by DARPA under Contract No. HR0011-12-C-0016. Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

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

Collins, M. 2002b. Ranking algorithms for named entity extraction: Boosting and the voted perceptron. In *ACL*, 489–496.

Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Singer, Y. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research* 7:551–585.

Dinarelli, M.; Moschitti, A.; and Riccardi, G. 2012. Discriminative reranking for spoken language understanding. *Audio, Speech, and Language Processing, IEEE Transactions on* 20(2):526 –539.

Finkel, J. R.; Grenager, T.; and Manning, C. D. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*.

Jin, G., and Chen, X. 2008. The fourth international chinese language processing bakeoff: Chinese word segmentation, named entity recognition and chinese pos tagging. In *Proceedings of Sixth SIGHAN Workshop on Chinese Language Processing*, 69–81.

Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 282–289.

Qian, X.; Jiang, X.; Zhang, Q.; Huang, X.; and Wu, L. 2009. Sparse higher order conditional random fields for improved sequence labeling. In *ICML*, 107.

Roth, D., and Yih, W.-T. 2005. Integer linear programming inference for conditional random fields. In *ICML*, 736–743.

Rush, A. M.; Sontag, D.; Collins, M.; and Jaakkola, T. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 1–11. Cambridge, MA: Association for Computational Linguistics.

Sarawagi, S., and Cohen, W. W. 2004. Semi-markov conditional random fields for information extraction. In *NIPS*.

Taskar, B.; Guestrin, C.; and Koller, D. 2003. Max-margin markov networks. In *NIPS*.

Weiss, D.; Sapp, B.; and Taskar, B. 2010. Sidestepping intractable inference with structured ensemble cascades. In *NIPS*, 2415–2423.

Wu, X.; Lin, X.; Wang, X.; Wu, C.; Zhang, Y.; and Yu, D. 2008. An improved crf based chinese language processing system for sighan bakeoff 2007. In *Proceedings of Sixth SIGHAN Workshop on Chinese Language Processing*.

Ye, N.; Lee, W. S.; Chieu, H. L.; and Wu, D. 2009. Conditional random fields with high-order features for sequence labeling. In *NIPS*, 2196–2204.