Multilayer Sequence Labeling

Ai Azuma **Yuji Matsumoto**
Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan
{ai-a,matsu}@is.naist.jp

Abstract

In this paper, we describe a novel approach to cascaded learning and inference on sequences. We propose a weakly joint learning model on cascaded inference on sequences, called multilayer sequence labeling. In this model, inference on sequences is modeled as cascaded decision. However, the decision on a sequence labeling sequel to other decisions utilizes the features on the preceding results as marginalized by the probabilistic models on them. It is not novel itself, but our idea central to this paper is that the probabilistic models on succeeding labeling are viewed as indirectly depending on the probabilistic models on preceding analyses. We also propose two types of efficient dynamic programming which are required in the gradient-based optimization of an objective function. One of the dynamic programming algorithms resembles back propagation algorithm for multilayer feed-forward neural networks. The other is a generalized version of the forward-backward algorithm. We also report experiments of cascaded part-of-speech tagging and chunking of English sentences and show effectiveness of the proposed method.

1 Introduction

Machine learning approach is widely used to classify instances into discrete categories. In many tasks, however, some set of inter-related labels should be decided simultaneously. Such tasks are called structured prediction. Sequence labeling is the simplest subclass of structured prediction problems. In sequence labeling, the most likely one among all the possible label sequences is predicted for a given input. Although sequence labeling is the simplest subclass, a lot of real-world tasks are modeled as problems of this simplest subclass. In addition, it might offer valuable insight and a toehold for more general and complex structured prediction problems. Many models have been proposed for sequence labeling tasks, such as Hidden Markov Models (HMM), Conditional Random Fields (CRF) (Lafferty et al., 2001), Max-Margin Markov Networks (Taskar et al., 2003) and others. These models have been applied to lots of practical tasks in natural language processing (NLP), bioinformatics, speech recognition, and so on. And they have shown great success in recent years.

In real-world tasks, it is often needed to cascade multiple predictions. A cascade of predictions here means the situation in which some of predictions are made based upon the results of other predictions. Sequence labeling is not an exception. For example, in NLP, we perform named entity recognition or base-phrase chunking for given sentences based on part-of-speech (POS) labels predicted by another sequence labeler. Natural languages are especially interpreted to have a hierarchy of sequential structures on different levels of abstraction. Therefore, many tasks in NLP are modeled as a cascade of sequence predictions.

If a prediction is based upon the result of another prediction, we call the former upper stage and the latter lower stage.

Methods pursued for a cascade of predictions – including sequence predictions, of course–, are desired to perform certain types of capability. One desired capability is rich forward information propagation, that is, the learning and estimation on each stage of predictions should utilize rich information of the results of lower stages whenever possible. "Rich information" here includes next bests and confidence information of the results of lower stages. Another is backward information propagation, that is, the rich annotated data on an upper stage should affect the models on lower stages retroactively.

Many current systems for a cascade of sequence predictions adopt a simple 1-best feed-forward approach. They simply take the most likely output at each prediction stage and transfer it to the next upper stage. Such a framework can maximize reusability of existing sequence labeling systems. On the other hand, it exhibits a strong tendency to propagate errors to upper labelers.

Typical improvement on the 1-best approach is to keep k-best results in the cascade of predictions. However, the larger k becomes, the more difficult it is to enumerate and maintain the k-best results. It is particularly prominent in sequence labeling.

The essence of this orientation is that the labeler on an upper stage utilizes the information of all the possible output candidates on lower stages. However, the size of the output space can become quite large in sequence labeling. It effectively forbids explicit enumeration of all possible outputs, so it is required to represent all the labeling possibilities compactly or employ some approximation schemes. Several studies are in this direction. In the method proposed in Finkel et al. (2006), a cascades of sequence predictions is viewed as a Bayesian network, and sample sequences are drawn at each stage according to the output distribution. The samples are then used to estimate the entire distribution of the cascade. In the method proposed in Bunescu (2008), an upper labeler uses the probabilities marginalized on the parts of the output sequences on lower stages as weights for the features. The weighted features are integrated in the model of the labeler on the upper stage. A k-best approach (e.g., (Collins and Duffy, 2002)) and the methods mentioned above are effective to improve the forward information propagation. However, they can never contribute on backward information propagation.

To improve the both directions of information propagation, Some studies propose the joint learning of multiple sequence labelers. Sutton et al. (2007) proposes the joint learning method in case where multiple labels are assigned to each time slice of the input sequences. It enables simultaneous learning and estimation of multiple sequence labelings on the same input sequences, where time slices of the outputs of all the out sequences are regularly aligned. However, it puts the distribution of states into Bayesian networks with cyclic dependencies, and exact inference is not tractable in such a model in general. Therefore, it requires some approximate inference algorithms in learning or predictions. Moreover, it only considers the cases where labels of an input sequence and all output sequences are regularly aligned. It is not clear how to build a joint labeling model which handles irregular output label sequences like semi-Markov models (Sarawagi and Cohen, 2005).

In this paper, we propose a middle ground for a cascade of sequence predictions. The proposed method adopts the basic idea of Bunescu (2008). We first assume that the model on all the sequence labeling stages is probabilistic one. In modeling of an upper stage, a feature is weighted by the marginal probability of the fragment of the outputs from a lower stage. However, this is not novel itself because it is just a paraphrase of Bunescu's core idea. Our intuition behind the proposed method is as follows. Features integrated in the model on each stage are weighted by the marginal probabilities of the fragments of the outputs on lower stages. So, if the output distributions on lower stages change, the marginal probabilities of any fragments also change, and this in turn can change the value of the features on the upper stage. In other words, the features on an upper stage indirectly depend on the models on the lower stages. Based on this intuition, the learning procedure of the model on an upper stage can affect not only direct model parameters, but also the weights of the features by changing the model on the lower stages. Supervised learning based on annotated data on an upper stage may affect the model or model parameters on the lower stages. It could be said that the information of annotation data on an upper stage is propagated back to the model on lower stages.

In the next section, we describe the formal notation of our model. In Section 3, we propose an optimization procedure according to the intuition noted above. In Section 4, we report an experimental result of our method. The proposed method shows some improvements on a real-world task in comparison with ordinary methods.

2 Formalization

In this section, we introduce the formal notation of our model. Hereafter, for the sake of simplicity, we only describe the simplest case in which there are just two stages, one lower stage of sequence labeling named L_1 and one upper stage of sequence labeling named L_2. In L_1, the most likely one among a set of possible sequences is predicted for a given input $\mathbf{x}$. L_2 is also a sequence labeling stage for the same input $\mathbf{x}$ and the output of L_1. No assumption is made on the structure of $\mathbf{x}$. The information of $\mathbf{x}$ is totally encoded in feature functions. It is only assumed that the output spaces of both L_1 and L_2 are conditioned on the initial input $\mathbf{x}$.

First of all, we describe the formalization of the probabilistic model for L_1. The model for L_1 per se is the same as ordinary ones for sequence labeling. For a given input $\mathbf{x}$, consider a directed acyclic graph (DAG) $G_1 = (V_1, E_1)$. A source of a DAG G is a node whose in-degree is equal to zero. A sink of a DAG G is nodes whose out-degree is equal to zero. Let $\mathrm{src}(G)$, $\mathrm{snk}(G)$ denote the set of source and sink nodes in G, respectively. A successful path of a DAG G is defined as a directed path on G whose starting node is a source and end node is a sink. If $\mathbf{y}$ denotes a path on a DAG, let $\mathbf{y}$ also denote the set of all the arcs appearing on $\mathbf{y}$ for the sake of shorthand. We denote the set of all the possible successful paths on G_1 by $\mathbf{Y}_1$. The space of the output candidates for L_1 is exactly equal to $\mathbf{Y}_1$. For the modeling of L_1, it is assumed that features of the form $f_{\langle 1,k_1,e_1,\mathbf{x}\rangle} \in \mathbb{R}$ $(k_1 \in \mathcal{K}_1, e_1 \in E_1)$ are allowed to be used. Here, $\mathcal{K}_1$ is the index set of the feature types for L_1. Such a feature can capture an aspect of the correlation between adjacent nodes. We call this kind of features input features for L_1. This naming is used to distinguish them from another kind of features defined on L_1, which comes later. Although features on V_1 can be also defined, they are totally omitted in this paper for brevity. Hereafter, if a symbol has subscripts, then missing subscript indicates a set that range over the omitted subscript. For example, $\mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle} \overset{\mathrm{def}}{\equiv} \left\{f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}\right\}_{k_1\in\mathcal{K}_1}$, $\mathbf{f}_{\langle 1,k_1,\mathbf{x}\rangle} \overset{\mathrm{def}}{\equiv} \left\{f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}\right\}_{e_1\in E_1}$, $\mathbf{f}_{\langle 1,\mathbf{x}\rangle} \overset{\mathrm{def}}{\equiv} \left\{f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}\right\}_{k_1\in\mathcal{K}_1,e_1\in E_1}$, and so on. The probabilistic model on L_1 forms the log-linear model, that is,

$$P_1(\mathbf{y}_1|\mathbf{x};\boldsymbol{\theta}_1) \overset{\mathrm{def}}{\equiv} \frac{1}{Z_1(\mathbf{x};\boldsymbol{\theta}_1)} \exp\big(\boldsymbol{\theta}_1 \cdot \mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\big) \quad (\mathbf{y}_1 \in \mathbf{Y}_1) \quad , \tag{1}$$

where $\theta_{\langle 1,k_1\rangle} \in \mathbb{R}$ $(k_1 \in \mathcal{K}_1)$ is the weight for the feature of the same index k_1, and the k_1-th element of $\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}$, $F_{\langle 1,k_1,\mathbf{y}_1,\mathbf{x}\rangle} \overset{\mathrm{def}}{\equiv} \sum_{e_1\in\mathbf{y}_1} f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}$. Dot operator $(\cdot)$ denotes the inner product with respect to the subscripts commonly missing in both operands. Z_1 is the partition function for P_1, defined as

$$Z_1(\mathbf{x};\boldsymbol{\theta}_1) \overset{\mathrm{def}}{\equiv} \sum_{\mathbf{y}_1\in\mathbf{Y}_1} \exp\big(\boldsymbol{\theta}_1 \cdot \mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\big) \quad . \tag{2}$$

It is worth noting that this formalization subsumes both directed and undirected linear-chain graphical models, which are the most typical models for sequence labeling, including HMM and CRF. That is, if the elements of V_1 are aligned into regular time slices, and the nodes in each time slice are associated with possible assignments of labels for that time, we obtain the representation equivalent to the ordinary linear-chain graphical models, in which all possible label assignments for each state are expanded. In such configuration, all the possible successful paths defined in our notation have strict one-to-one correspondence to all the possible joint assignments of labels in linear-chain graphical models. We purposely employ this DAG-based notation because; it is convenient to describe the models and algorithms for our purpose, it allows for labels to stay in arbitrary time as in semi-Markov models, and it is easily extended to models for a set of trees instead of sequences by replacing the graph-based notation with hypergraph-based notation.

Next, we formalize the probabilistic model on the upper stage L_2. Like L_1, consider a DAG $G_2 = (V_2, E_2)$ conditioned on the input $\mathbf{x}$, and the set of all the possible successful paths on G_2, denoted $\mathbf{Y}_2$. The space of the output candidates for L_2 becomes $\mathbf{Y}_2$.

The form of the features available in designing the probabilistic model for L_2, denoted by P_2, is the key of this paper. A feature on an arc $e_2 \in E_2$ can access local characteristics of the confidence-rated superposition of the L_1's outputs, in addition to the information of the input $\mathbf{x}$. To formulate local characteristics of the superposition of the L_1's outputs, we first define output features of L_1, denoted by $h_{\langle 1,k'_1,e_1\rangle} \in \mathbb{R} \quad (k'_1 \in \mathcal{K}'_1,\, e_1 \in E_1)$. Here, $\mathcal{K}'_1$ is the index set of the output feature types of L_1. Before the output features are integrated into the model for L_2, they all are confidence-rated with respect to P_1, that is, each output feature $h_{\langle 1,k'_1,e_1\rangle}$ is numerically rated by the estimated probabilities summed over the sequences emitting that feature. More formally, all the L_1's output features are integrated in features for P_2 in the form of the marginalized output features, which are defined as follows;

$$\bar{h}_{\langle 1,k'_1,e_1\rangle}(\boldsymbol{\theta}_1) \overset{\text{def}}{\equiv} h_{\langle 1,k'_1,e_1\rangle} P_1(e_1|\mathbf{x};\boldsymbol{\theta}_1) \quad \left(k'_1 \in \mathcal{K}'_1,\, e_1 \in E_1\right) \quad , \tag{3}$$

where

$$\begin{aligned} P_1(e_1|\mathbf{x};\boldsymbol{\theta}_1) &\overset{\text{def}}{\equiv} \sum_{\mathbf{y}_1 \sim e_1} P_1(\mathbf{y}_1|\mathbf{x};\boldsymbol{\theta}_1) \\ &= \sum_{\mathbf{y}_1 \in \mathbf{Y}_1} \delta_{e_1 \in \mathbf{y}_1} P_1(\mathbf{y}_1|\mathbf{x};\boldsymbol{\theta}_1) \\ &\qquad (e_1 \in E_1) \quad . \end{aligned} \tag{4}$$

Here, the notation $\sum_{\mathbf{y}_1 \sim e_1}$ represents the summation over sequences consistent with an arc $e_1 \in E_1$, that is, the summation over the set $\{\mathbf{y}_1 \in \mathbf{Y}_1 \mid e_1 \in \mathbf{y}_1\}$. $\delta_{\mathcal{P}}$ denotes the indicator function for a predicate $\mathcal{P}$. The input features for P_2 on an arc $e_2 \in E_2$ are permitted to arbitrarily combine the information of $\mathbf{x}$ and the L_1's marginalized output features $\bar{\mathbf{h}}_1$, in addition to the local characteristics of the arc at hand e_2. In summary, an input feature for L_2 on an arc $e_2 \in E_2$ is of the form

$$f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}\left(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\right) \in \mathbb{R} \qquad (k_2 \in \mathcal{K}_2) \quad , \tag{5}$$

where $\mathcal{K}_2$ is the index set of the input feature types for L_2. To make the optimization procedure feasible, smoothness condition on any L_2's input feature is assumed with respect to all the L_1's output features, that is, $\frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}}{\partial \bar{h}_{\langle 1,k'_1,e_1\rangle}}$ is always guaranteed to exist for $^\forall k'_1,\, e_1,\, k_2,\, e_2$. For example, additions and multiplications between some elements of $\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)$ can appear in the definition of L_2's input features. For given input features $\mathbf{f}_{\langle 2,\mathbf{x}\rangle}\left(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\right)$ and parameters $\theta_{\langle 2,k_2\rangle} \in \mathbb{R} \quad (k_2 \in \mathcal{K}_2)$, the probabilistic model for L_2 is defined as follows;

$$\begin{aligned} &P_2(\mathbf{y}_2|\mathbf{x};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) \\ &\quad \overset{\text{def}}{\equiv} \frac{1}{Z_2(\mathbf{x};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)} \exp\left(\boldsymbol{\theta}_2 \cdot \mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\left(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\right)\right) \\ &\qquad (\mathbf{y}_2 \in \mathbf{Y}_2) \quad , \end{aligned} \tag{6}$$

where $F_{\langle 2,k_2,\mathbf{y}_2,\mathbf{x}\rangle}\left(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\right) \overset{\text{def}}{\equiv} \sum_{e_2 \in \mathbf{y}_2} f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}\left(\bar{\mathbf{h}}_1\left(\boldsymbol{\theta}_1\right)\right)$ and Z_2 is the partition function of P_2, defined by

$$\begin{aligned} &Z_2(\mathbf{x};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) \\ &\quad \overset{\text{def}}{\equiv} \sum_{\mathbf{y}_2 \in \mathbf{Y}_2} \exp\left(\boldsymbol{\theta}_2 \cdot \mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\left(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\right)\right) \quad . \end{aligned} \tag{7}$$

The definition of P_2 (6) reveals one of the most important points in this paper. P_2 is viewed not only as the function of the ordinary direct parameters $\boldsymbol{\theta}_2$ but also as the function of $\boldsymbol{\theta}_1$, which represents the parameters for the L_1's model, through the intermediate variables $\bar{\mathbf{h}}_1$. So optimization procedure on P_2 may affect the determination of the values not only of the direct parameters $\boldsymbol{\theta}_2$ but also of the indirect ones $\boldsymbol{\theta}_1$.

If the result of L_1 is reduced to the single golden output $\tilde{\mathbf{y}}_1$, i.e. $P_1(\mathbf{y}_1|\mathbf{x}) = \delta_{\mathbf{y}_1 = \tilde{\mathbf{y}}_1}$, the definitions above boil down to the formulation of the simple 1-best feed forward architecture.

3 Optimization Algorithm

In this section, we describe optimization procedure for the model formulated in the previous section. Let $\mathcal{D} = \{\langle \hat{\mathbf{x}}, \langle G_1, \hat{\mathbf{y}}_1\rangle, \langle G_2, \hat{\mathbf{y}}_2\rangle\rangle_m\}_{m=1,2,\cdots,M}$ denote annotated data for the supervised learning of the model. Here, $\langle G_1, \hat{\mathbf{y}}_1\rangle$ is a pair of a DAG and correctly annotated successful sequence for L_1. The same holds for $\langle G_2, \hat{\mathbf{y}}_2\rangle$. For given $\mathcal{D}$, we can define the conditional log-likelihood function on L_1 and L_2 respectively, that is,

$$\begin{aligned} &\mathcal{L}_1\left(\boldsymbol{\theta}_1;\mathcal{D}\right) \\ &\quad \overset{\text{def}}{\equiv} \sum_{\langle \hat{\mathbf{x}},\hat{\mathbf{y}}_1\rangle \in \mathcal{D}} \log\left(P_1\left(\hat{\mathbf{y}}_1|\hat{\mathbf{x}};\boldsymbol{\theta}_1\right)\right) - \frac{|\boldsymbol{\theta}_1|}{2{\sigma_1}^2} \end{aligned} \tag{8}$$

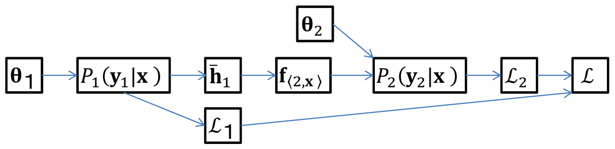

Figure 1: Computation Graph of the Proposed Model

and

$$\mathcal{L}_2\left(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; \mathcal{D}\right) \overset{\text{def}}{\equiv} \sum_{\langle\hat{\mathbf{x}},\hat{\mathbf{y}}_2\rangle\in\mathcal{D}} \log\left(P_2\left(\hat{\mathbf{y}}_2|\hat{\mathbf{x}}; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2\right)\right) - \frac{|\boldsymbol{\theta}_2|}{2{\sigma_2}^2} \ . \tag{9}$$

Here, ${\sigma_1}^2$, ${\sigma_2}^2$ are the variances of the prior distributions of the parameters. For the sake of simplicity, we set the prior distribution as the zero-mean univariance Gaussian. To optimize the both probabilistic models P_1 and P_2 jointly, we also define the joint conditional log-likelihood function

$$\mathcal{L}\left(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; \mathcal{D}\right) \overset{\text{def}}{\equiv} \mathcal{L}_1 + \mathcal{L}_2 \ . \tag{10}$$

The parameter values to be learned are the ones that (possibly locally) maximize this objective function. Note that this objective function is not guaranteed to be globally convex.

We employ gradient-based parameter optimization here. Optimization procedure repeatedly searches a direction in the parameter space which is ascendent with respect to the objective function, and updates the parameter values into that direction by small advances. Many existing optimization routines like steepest descent or conjugation gradient do that job only by giving the objective value and gradients on parameter values to be updated. So, the optimization problem here boils down to the calculation of the objective value and gradients on given parameter values.

Before entering the detailed description of the algorithm for calculating the objective function and gradients, we note the functional relations among the objective function and previously defined variables. The diagram shown in Figure 1 illustrates the functional relations among the parameters, input and output feature functions, models, and objective function. The variables at the head of a directed arrow in the figure is directly defined in terms of the ones at the tail of the same arrow. The value of the objective function on given parameter values can be calculated in order of the arrows shown in the diagram. On the other hand, the parameter gradients are calculated step-by-step in reverse order of the arrows. The functional relations illustrated in the Figure 1 ensure some forms of the chain rule of differentiation among the variables. The chain rule is iteratively used to decompose the calculation of the gradients into a divide-and-conquer fashion. These two directions of stepwise computation are analogous to the forward and back propagation for multilayer feedforward neural networks, respectively.

Algorithm 1 shows the whole picture of the gradient-based optimization procedure for our model. We first describe the flow to calculate the objective value for a given parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, which is shown from line 2 through 4 in Algorithm 1. The values of marginalized output features $\bar{\mathbf{h}}_{\langle 1,\mathbf{x}\rangle}$ can be calculated by (3). Because they are the simple marginals of features, the ordinary forward-backward algorithm (hereafter, abbreviated as "F-B") on G_1 offers an efficient way to calculate their values. Although nothing definite about the forms of the input features for L_2 is presented in this paper, $\mathbf{f}_{\langle 2,\mathbf{x}\rangle}$ can be calculated once the values of $\bar{\mathbf{h}}_{\langle 1,\mathbf{x}\rangle}$ have been obtained. Finally, $\mathcal{L}_1$, $\mathcal{L}_2$ and then $\mathcal{L}$ are easy to calculate because they are no different from the ordinary log-likelihood computation.

Now we describe the algorithm to calculate the parameter gradients,

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}_1} = \frac{\partial\mathcal{L}_1}{\partial\boldsymbol{\theta}_1} + \frac{\partial\mathcal{L}_2}{\partial\boldsymbol{\theta}_1}, \qquad \frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}_2} = \frac{\partial\mathcal{L}_2}{\partial\boldsymbol{\theta}_2} \ . \tag{11}$$

Line 5 through line 7 in Algorithm 1 describe the gradient computation. The terms $\frac{\partial\mathcal{L}_1}{\partial\boldsymbol{\theta}_1}$ and $\frac{\partial\mathcal{L}_2}{\partial\boldsymbol{\theta}_2}$ in (11) become the same forms that appear in the ordinary CRF optimization, i.e., the difference between the empirical frequencies of the features and the model expectations of them,

$$\begin{aligned} \frac{\partial\mathcal{L}_1}{\partial\boldsymbol{\theta}_1} &= \tilde{E}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right] - E_{P_1}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right] - \frac{|\boldsymbol{\theta}_1|}{{\sigma_1}^2} \ , \\ \frac{\partial\mathcal{L}_2}{\partial\boldsymbol{\theta}_2} &= \tilde{E}\left[\mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\right] - E_{P_2}\left[\mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\right] - \frac{|\boldsymbol{\theta}_2|}{{\sigma_2}^2} \ . \end{aligned} \tag{12}$$

These calculations are performed by the ordinary F-B on G_1 and G_2, respectively. Using the chain rule of differentiation derived from the functional relations illustrated in Figure 1, the remaining term $\frac{\partial L_2}{\partial\boldsymbol{\theta}_1}$

Algorithm 1 Gradient-based optimization of the model parameters

```
Input: $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$
Output: $\underset{\langle\boldsymbol{\theta}_1,\boldsymbol{\theta}_2\rangle}{\arg\max}\,\mathcal{L}$
1: while $\boldsymbol{\theta}_1$ or $\boldsymbol{\theta}_2$ changes significantly do
2:     calculate $Z_1$ by (2), $\bar{\mathbf{h}}_1$ by (3) with the F-B on $G_1$, and then $\mathcal{L}_1$ by (8)
3:     calculate $\mathbf{f}_{\langle 2,\mathbf{x}\rangle}$ according to their definitions
4:     calculate $Z_2$ by (7) with the F-B on $G_2$, and then $\mathcal{L}_2$ by (9) and $\mathcal{L}$ by (10)
5:     calculate $\frac{\partial\mathcal{L}_1}{\partial\boldsymbol{\theta}_1}$ and $\frac{\partial\mathcal{L}_2}{\partial\boldsymbol{\theta}_2}$ by (12) with the F-B on $G_1$ and $G_2$, respectively
6:     calculate $\frac{\partial\mathcal{L}}{\partial\mathbf{f}_{\langle 1,\mathbf{x}\rangle}}$ by (16) with the F-B on $G_2$, $\frac{\partial\mathbf{f}_{\langle 1,\mathbf{x}\rangle}}{\partial\bar{\mathbf{h}}_1}$, and them $\frac{\partial\mathcal{L}_2}{\partial\bar{\mathbf{h}}_1} = \frac{\partial\mathcal{L}}{\partial\mathbf{f}_{\langle 1,\mathbf{x}\rangle}} \cdot \frac{\partial\mathbf{f}_{\langle 1,\mathbf{x}\rangle}}{\partial\bar{\mathbf{h}}_1}$
7:     calculate $\frac{\partial\mathcal{L}_2}{\partial\boldsymbol{\theta}_1}$ by (18) with Algorithm 2
8:     $\langle\boldsymbol{\theta}_1,\boldsymbol{\theta}_2\rangle \leftarrow$ update-parameters$\left(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\mathcal{L},\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}_1},\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}_2}\right)$
9: end while
```

in (11) can be decomposed as follows;

$$\frac{\partial\mathcal{L}_2}{\partial\boldsymbol{\theta}_1} = \frac{\partial\mathcal{L}_2}{\partial\mathbf{f}_{\langle 2,\mathbf{x}\rangle}} \cdot \frac{\partial\mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial\boldsymbol{\theta}_1} = \frac{\partial\mathcal{L}_2}{\partial\mathbf{f}_{\langle 2,\mathbf{x}\rangle}} \cdot \frac{\partial\mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial\bar{\mathbf{h}}_1} \cdot \frac{\partial\bar{\mathbf{h}}_1}{\partial\boldsymbol{\theta}_1} \ . \tag{13}$$

Note that Leibniz's notation here denotes a Jacobian with the index sets omitted in the numerator and the denominator, for example,

$$\frac{\partial\mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial\bar{\mathbf{h}}_1} \overset{\text{def}}{\equiv} \left\{\frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}}{\partial h_{\langle 1,k'_1,e_1\rangle}}\right\}_{k_2\in\mathcal{K}_2,e_2\in E_2,k'_1\in\mathcal{K}'_1,e_1\in E_1} \tag{14}$$

And also recall that dot operators here stand for the inner product with respect to the index sets commonly omitted in both operands, for example,

$$\frac{\partial\mathcal{L}_2}{\partial\mathbf{f}_2} \cdot \frac{\partial\mathbf{f}_2}{\partial\bar{\mathbf{h}}_1} = \sum_{k_2\in\mathcal{K}_2,e_2\in E_2} \frac{\partial\mathcal{L}_2}{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}} \cdot \frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}}{\partial\bar{\mathbf{h}}_1} \ . \tag{15}$$

We describe the manipulation of each factor in the right side of (13) in turn. Noting $\frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}}{\partial f_{\langle 2,\grave{k}_2,\grave{e}_2,\mathbf{x}\rangle}} = \delta_{k_2=\grave{k}_2}\delta_{e_2=\grave{e}_2}$, each element of the first factor of (13) $\frac{\partial\mathcal{L}_2}{\partial\mathbf{f}_{\langle 2,\mathbf{x}\rangle}}$ can be transformed as follows;

$$\frac{\partial\mathcal{L}_2}{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}} = \theta_{\langle 2,k_2\rangle} \sum_{\langle\hat{\mathbf{x}},\hat{\mathbf{y}}_2\rangle\in\mathcal{D}} \left(\delta_{e_2\in\hat{\mathbf{y}}_2} - P_2(e_2|\hat{\mathbf{x}};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)\right) \ . \tag{16}$$

$P_2(e_2|\hat{\mathbf{x}};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)$, the marginal probability on e_2, can be obtained as a by-product of the F-B for (12).

As described in the previous section, it is assumed that the values of the second factor $\frac{\partial\mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial\bar{\mathbf{h}}_1}$ is guaranteed to exists for any given $\boldsymbol{\theta}_1$, and the procedure for calculating them is fixed in advance. The procedure for some of concrete features is exemplified in the previous section.

From the definition of $\bar{\mathbf{h}}_1$ (3), each element of the third factor of (13) $\frac{\partial\bar{\mathbf{h}}_1}{\partial\boldsymbol{\theta}_1}$ becomes

$$\frac{\partial\bar{h}_{\langle 1,k'_1,e_1\rangle}}{\partial\theta_{\langle 1,k_1\rangle}} = h_{\langle 1,k'_1,e_1\rangle}\mathrm{Cov}_{P_1(\mathbf{y}_1|\mathbf{x})}\left[\delta_{e_1\in\mathbf{y}_1},\, F_{\langle 1,k_1,\mathbf{y}_1,\mathbf{x}\rangle}\right] \ . \tag{17}$$

There exists efficient dynamic programming to calculate the covariance value (17) (without goint into that detail because it is very similar to the one shown later in this paper), and of course we can run such dynamic programming for ${}^\forall k'_1 \in \mathcal{K}'_1$, $e_1 \in E_1$. However, the size of the Jacobian $\frac{\partial\bar{\mathbf{h}}_1}{\partial\boldsymbol{\theta}_1}$ is equal to $|\mathcal{K}'_1|\times|E_1|\times|\mathcal{K}_1|$. Since it is too large in many tasks likely to arise in practice, we should avoid to calculate all the elements of this Jacobian in a straightforward way. Instead of such naive computation, if the values of $\frac{\partial\mathcal{L}_2}{\partial\mathbf{f}_{\langle 2,\mathbf{x}\rangle}}$ and $\frac{\partial\mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial\bar{\mathbf{h}}_1}$ are obtained, then we can compute $\frac{\partial\mathcal{L}_2}{\partial\bar{\mathbf{h}}_1} = \frac{\partial\mathcal{L}_2}{\partial\mathbf{f}_{\langle 2,\mathbf{x}\rangle}} \cdot \frac{\partial\mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial\bar{\mathbf{h}}_1}$, and from (13) and (17),

$$\begin{aligned}\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_1} &= \frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_1} \cdot \frac{\partial \bar{\mathbf{h}}_1}{\partial \boldsymbol{\theta}_1} \\ &= E_{P_1(\mathbf{y}_1|\mathbf{x})}\left[H'_{\langle 1,\mathbf{y}_1\rangle}\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right] \\ &\quad - E_{P_1(\mathbf{y}_1|\mathbf{x})}\left[H'_{\langle 1,\mathbf{y}_1\rangle}\right] E_{P_1(\mathbf{y}_1|\mathbf{x})}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right] \quad ,\end{aligned} \tag{18}$$

where $H'_{\langle 1,\mathbf{y}_1\rangle} \overset{\text{def}}{\equiv} \sum_{e_1 \in \mathbf{y}_1} \frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_{\langle 1,e_1\rangle}} \cdot \mathbf{h}_{\langle 1,e_1\rangle}$. In other words, $\frac{\partial \mathcal{L}_2}{\partial \theta_{\langle 1,k_1\rangle}}$ becomes the covariance between the k_1-th input feature for L_1 and the hypothetical feature $h'_{\langle 1,e_1\rangle} \overset{\text{def}}{\equiv} \frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_{\langle 1,e_1\rangle}} \cdot \mathbf{h}_{\langle 1,e_1\rangle}$.

The final problem is to derive an efficient way to compute the first term of (18). The second term of (18) can be calculated by the ordinary F-B because it consists of the marginals of arc features. There are two derivations of the algorithm for calculating the first term. We describe briefly the both derivations.

One is a variant of the F-B on the expectation semi-ring proposed in Li and Eisner (2009). First, the F-B is generalized to the expectation semi-ring with respect to the hypothetical feature $h'_{\langle 1,e_1\rangle}$, and by summing up the marginals of the feature vectors $\mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle}$ on all the arcs under the distribution of the semi-ring, then we obtain the expectation of the feature vector $\mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle}$ on the semi-ring potential. This expectation is equal to the first term of (18). [1]

Another derivation is to apply the automatic differentiation (AD)(Wengert, 1964; Corliss et al., 2002) on the F-B calculating $E_{P_1}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right]$. It exploits the fact that $\frac{\partial}{\partial \lambda} E_{P'_1}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right]\Big|_{\lambda=0}$ is equal to the first term of (18), where $\lambda \in \mathbb{R}$ is a dummy parameter, and $P'_1(\mathbf{y}_1|\mathbf{x}) \overset{\text{def}}{\equiv} \frac{1}{Z_1}\exp\Big(\boldsymbol{\theta}_1 \cdot \mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle} + \lambda H'_{\langle 1,\mathbf{y}_1\rangle}\Big)$. It is easy to derive the F-B for calculating the value $E_{P'_1}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right]\Big|_{\lambda=0}$. AD transforms this F-B into another algorithm for calculating the differentiation w.r.t. λ evaluated at the point $\lambda = 0$. This transformation is achieved in an automatic manner, by replacing all appearances of λ in the F-B with a dual number $\lambda + \varepsilon$. The dual number is a variant of the complex number, with a kind of the imaginary unit ε with the property $\varepsilon^2 = 0$. Like the usual complex numbers, the arithmetic operations and the exponential function are generalized to the dual numbers, and the ordinary F-B is also generalized to the dual numbers. The imaginary part of the resulting values is equal to the needed differentiation. [2] Anyway, these two derivations lead to the same algorithm, and the resulting algorithm is shown as Algorithm 2.

The final line in the loop of Algorithm 1 can be implemented by various optimization routines and line search algorithms.

The time and space complexity to compute the objective and gradient values for given parameter vectors $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ is the same as that for that for Bunescu (2008), up to a constant factor. Because the calculation of the objective function is essentially the same as that for Bunescu (2008), and in gradient computation, the time complexity of Algorithm 1 is the same as that for the ordinary F-B (up to a constant factor), and the proposed optimization procedure is only required to store additional scalar values $h'_{\langle 1,e_1\rangle}$ on each G_1's arc.

4 Experiment

We examined effectiveness of the method proposed in this paper on a real task. The task is to annotate the POS tags and to perform base-phrase chunking on English sentences.

Base-phrase chunking is a task to classify continuous subsequences of words into syntactic categories. This task is performed by annotating a chunking label on each word (Ramshaw and Marcus, 1995). The types of chunking label consist of "Begin-*Category*", which represents the beginning of a chunk, "Inside-*Category*", which represents the inside of a chunk, and "Other." Usually, POS labeling runs first before base-phrase chunking is performed. Therefore, this task is a typical interesting case where a sequence labeling depends on the output from other sequence labelers.

The data used for our experiment consist of English sentences from the Penn Treebank project (Marcus et al., 1993) consisting of 10948 sentences and 259104 words. We divided them into two groups, training data consisting of 8936 sentences and 211727 words and test data consisting of 2012

[1] For the detailed description, see Li and Eisner (2009) and its references.

[2] For example, Berz (1992) gives a detailed description of the reason why the dual number is used for this purpose.

Algorithm 2 Forward-backward Algorithm for Calculating Feature Covariances

```
Input: $\mathbf{f}_{\langle 1,\mathbf{x}\rangle},\ \phi_{e_1} \overset{\mathrm{def}}{\equiv} \exp\big(\boldsymbol{\theta}_1 \cdot \mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle}\big),\ h'_{e_1} \overset{\mathrm{def}}{\equiv} \frac{\partial \mathcal{L}_2}{\partial \mathbf{h}_{\langle 1,e_1\rangle}} \cdot \mathbf{h}_{\langle 1,e_1\rangle}$
Output: $q_{k_1} = \mathrm{Cov}_{P(\mathbf{y}_1|\mathbf{x})}\left[H'_{\langle 1,\mathbf{y}_1\rangle},\ F_{\langle 1,k_1,\mathbf{y}_1,\mathbf{x}\rangle}\right] \quad \big({}^{\forall}k_1 \in \mathcal{K}_1\big)$
 1: for ${}^{\forall}v_1 \in \mathrm{src}(G_1),\ \alpha_{v_1} \leftarrow 1,\ \alpha'_{v_1} \leftarrow 1$
 2: for all $v_1 \in V_1$ in a topological order do
 3:     prev $\leftarrow \{x \in V_1 \mid (x, v_1) \in E_1\}$
 4:     $\alpha_{v_1} \leftarrow \sum_{x \in \mathrm{prev}} \phi_{(x,v_1)} \alpha_x,\ \alpha'_{v_1} \leftarrow \sum_{x \in \mathrm{prev}} \phi_{(x,v_1)} \left(h'_{(x,v_1)} \alpha_x + \alpha'_x\right)$
 5: end for
 6: $Z_1 \leftarrow \sum_{x \in \mathrm{snk}(G_1)} \alpha_x$
 7: for ${}^{\forall}v_1 \in \mathrm{snk}(G_1),\ \beta_{v_1} \leftarrow 1,\ \beta'_{v_1} \leftarrow 1$
 8: for all $v_1 \in V_1$ in a reverse topological order do
 9:     next $\leftarrow \{x \in V_1 \mid (v_1, x) \in E_1\}$
10:     $\beta_{v_1} \leftarrow \sum_{x \in \mathrm{next}} \phi_{(v_1,x)} \beta_x,\ \beta'_{v_1} \leftarrow \sum_{x \in \mathrm{next}} \phi_{(v_1,x)} \left(h'_{(v_1,x)} \beta_x + \beta'_x\right)$
11: end for
12: for ${}^{\forall}k_1 \in \mathcal{K}_1,\ q_{k_1} \leftarrow 0$
13: for all $(u_1, v_1) \in E_1$ do
14:     $p \leftarrow \phi_{(u_1,v_1)} \left(\alpha_{u_1} \beta'_{v_1} + \alpha'_{u_1} \beta_{v_1}\right) / Z_1$
15:     for ${}^{\forall}k_1 \in \mathcal{K}_1,\ q_{k_1} \leftarrow q_{k_1} + p f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}$
16: end for
```

sentences and 47377 words. The number of the POS label types is equal to 45. The number of the label types used in base-phrase chunking is equal to 23.

We compare the proposed method to two existing sequence labeling methods as baselines. The POS labeler is the same in all the three methods used in this experiment. This labeler is a simple CRF and learned by ordinary optimization procedure. One baseline method is the 1-best pipeline method. A simple CRF model is learned for the chunking labeling, on the input sentences and the most likely POS label sequences predicted by the already learned POS labeler. We call this method "CRF + CRF." The other baseline method has a CRF model for the chunking labeling, which uses the marginalized features offered by the POS labeler. However, the parameters of the POS labeler are fixed in the training of the chunking model. This method corresponds to the method proposed in Bunescu (2008). We call this baseline "CRF + CRF-MF" ("MF" for "marginalized features"). The proposed method is the same as "CRF + CRF-MF", except that the both labelers are jointly trained by the procedure described in Section 3. We call this proposed method "CRF + CRF-BP" ("BP" for "back propagation").

	CRF + CRF	CRF + CRF-MF	CRF +CRF-BP
POS labeling	95.6	(95.6)	95.8
Base-phrase chunking	92.1	92.7	**93.1**

Table 2: Experimental result (F-measure)

In "CRF + CRF-BP," the objective function for joint learning (10) is not guaranteed to be convex, so optimization procedure is sensible to the initial configuration of the model parameters. In this experiment, we set the parameter values learned by "CRF + CRF-MF" as the initial values for the training of the "CRF + CRF-BP" method. Feature templates used in this experiment are listed in Table 1. Although we only described the formalization and optimization procedure of the models with arc features, We use node features in the experiment.

Table 2 shows the result of the methods we men-

=== Node feature templates ===
Node is source
Node is sink
Input word on the same time slice
Suffix of input word on the same time slice, n characters $(n \in [1, 2, 3])$
Initial word character is capitalized†
All word characters are capitalized†
Input word included in the vocabulary of POS $T^\dagger$ $(T \in \{(\text{All possible POS labels})\})$
Input word contains numbers†
POS label‡
=== Arc feature templates ===
Tail node is source
Head node is sink
Corresponding ordered pair of POS labels‡

Table 1: List of feature templates. All node features are combined with the corresponding node label (POS or chunking label) feature. All arc features are combined with the feature of the corresponding arc label pair. † features are instantiated on each time slice in five character window. ‡ features are not used in POS labeler, and marginalized as output features for “CRF + CRF-MF” and “CRF + CRF-BP.”

tioned. In Table 2, bold numbers indicate significant improvement over the baseline models with $\alpha = 0.05$. From Table 2, the proposed method significantly outperforms two baseline methods on chunking performance. Although the improvement on POS labeling performance by the proposed method “CRF + CRF-BP” is not significant, it might show that optimization procedure provides some form of backward information propagation in comparison to “CRF + CRF-MF.”

5 Conclusions

In this paper, we adopt the method to weight features on an upper sequence labeling stage by the marginalized probabilities estimated by the model on lower stages. We also point out that the model on an upper stage is considered to depend on the model on lower stages indirectly. In addition, we propose optimization procedure that enables the joint optimization of the multiple models on the different level of stages. We perform an experiment on a real-world task, and our method significantly outperforms existing methods.

We examined the effectiveness of the proposed method only on one task in comparison to just a few existing methods. In the future, we hope to compare our method to other competing methods like joint learning approaches in terms of both accuracy and computational efficiency, and perform extensive experiments on various tasks.

References

M. Berz. 1992. Automatic differentiation as nonarchimedean analysis. In *Computer Arithmetic and Enclosure*, pages 439–450.

R.C. Bunescu. 2008. Learning with probabilistic features for improved pipeline models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 670–679.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics.

G.F. Corliss, C. Faure, and A. Griewank. 2002. *Automatic differentiation of algorithms: from simulation to optimization*. Springer Verlag.

J.R. Finkel, C.D. Manning, and A.Y. Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Z. Li and J. Eisner. 2009. First-and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 40–51.

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):330.

L.A. Ramshaw and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pages 82–94. Cambridge MA, USA.

S. Sarawagi and W.W. Cohen. 2005. Semi-markov conditional random fields for information extraction. *Advances in Neural Information Processing Systems*, 17:1185–1192.

C. Sutton, A. McCallum, and K. Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *The Journal of Machine Learning Research*, 8:693–723.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*.

RE Wengert. 1964. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):464.