A Generalization of Forward-Backward Algorithm

Ai Azuma and Yuji Matsumoto

Nara Institute of Science and Technology
{ai-a,matsu}@is.naist.jp

Abstract. Structured prediction has become very important in recent years. A simple but notable class of structured prediction is one for sequences, so-called sequential labeling. For sequential labeling, it is often required to take a summation over all the possible output sequences, when estimating the parameters of a probabilistic model for instance. We cannot make the direct calculation of such a summation from its definition in practice. Although the ordinary forward-backward algorithm provides an efficient way to do it, it is applicable to limited types of summations. In this paper, we propose a generalization of the forward-backward algorithm, by which we can calculate much broader types of summations than the existing forward-backward algorithms. We show that this generalization subsumes some existing calculations required in past studies, and we also discuss further possibilities of this generalization.

1 Introduction

Many learning tasks in the real world involve complex structures, where there are multiple, interrelated labels to be assigned. A simple but notable class that involves structures is sequential labeling, where dependencies between labels constitute a linear chain. A lot of classification or pattern recognition tasks on natural language sentences, genes and the like can be formulated as sequential labeling.

For sequential labeling, it is often required to take a summation over all the possible output sequences. For example, the expectation-maximization (EM) algorithm for Hidden Markov Models (HMMs) [1] requires expected frequencies of hidden variables measured on the current model. Another example is the calculation of the normalization constant, or partition function, for Conditional Random Fields (CRFs) [2]. Gradient of the log-likelihood objective for CRFs also involves model expectations of features. These calculations are originally defined as summations over all the possible output sequences.

Such a naive definition of a summation is not suitable for practical computing purposes in its own, because the number of the possible output sequences is proportional to the exponential of the length of the input sequence in general. The forward-backward algorithm provides a solution to get around this difficulty when output sequences are well-structured. "Well-structuredness" means that

W. Buntine et al. (Eds.): ECML PKDD 2009, Part I, LNAI 5781, pp. 99–114, 2009.

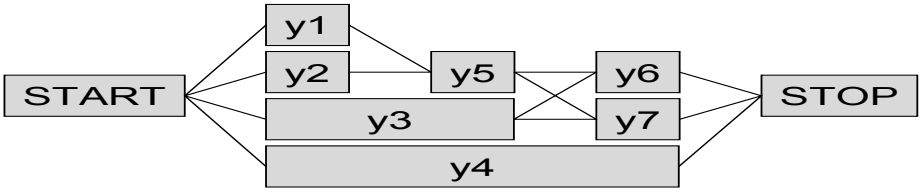

Fig. 1. An example trellis

output sequences can be encoded in a trellis, in which the number of nodes and arcs is proportional to a polynomial of the length of the input sequence. For example, the trellis shown in Fig.1 encodes seven sequences. This algorithm plays a crucial role in the aforementioned calculations. However, we would argue that the scope of applicability of the ordinary forward-backward algorithm is quite limited. The ordinary forward-backward algorithm is applicable to either one of the following forms,

$$\sum_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})}\left(\prod_{c\in\mathcal{C}(\boldsymbol{y})}\phi(\boldsymbol{x},c)\right)\ , \tag{1}$$

or

$$\sum_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})}\left(\prod_{c\in\mathcal{C}(\boldsymbol{y})}\phi(\boldsymbol{x},c)\right)\left(\sum_{c\in\mathcal{C}(\boldsymbol{y})}f(\boldsymbol{x},c)\right)\ , \tag{2}$$

where $\mathcal{Y}(\boldsymbol{x})$ denotes the set of all the possible sequences for a given input sequence $\boldsymbol{x}$, $\mathcal{C}(\boldsymbol{y})$ denotes the set of labels and adjacent pairs of labels appearing in $\boldsymbol{y}$, ϕ and f are complex-valued functions. Typically, ϕ represents a potential function of the distribution, and f represents an indicator function or a feature function.

We now want to derive an efficient algorithm applicable to much broader types of summations than (1) and (2). The term "efficient" here means that the computational cost is proportional not to the exponential of the length of the input sequence, but to a polynomial of it. As a result of the generalization proposed in this paper, we will derive an efficient algorithm applicable to the following form.

$$\sum_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})}\left(\prod_{c\in\mathcal{C}(\boldsymbol{y})}\phi(\boldsymbol{x},c)\right)\left(\sum_{c\in\mathcal{C}(\boldsymbol{y})}f_1(\boldsymbol{x},c)\right)^{n_1}\cdots\left(\sum_{c\in\mathcal{C}(\boldsymbol{y})}f_K(\boldsymbol{x},c)\right)^{n_K}\ , \tag{3}$$

where $f_1,\ldots,f_K$ are K complex-valued functions, and $n_1,\ldots,n_K$ are non-negative integers.

In the next section, we will formalize our algorithm and show its validity. We will also give arguments in support of our estimation of its computational cost. In Sect.3, we will show that some specializations of the proposed algorithm lead

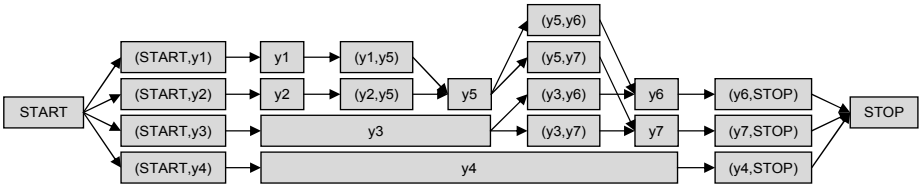

Fig. 2. The directed acyclic graph for the trellis shown in Fig.1

to some specific algorithms introduced in past studies. In Sect.4, we will show further possibilities of the algorithm. In Sect.5, we conclude this paper.

2 A Generalization of Forward-Backward Algorithm

In this section, we will formalize a generalization of the forward-backward algorithm. In order to write down definitions and proofs, we utilize directed acyclic graphs (DAG) instead of trellises. Consider a DAG $G = (V, E)$. A node in G represents either a label or an adjacent label pair in the trellis, i.e., $V := \bigcup_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \mathcal{C}(\boldsymbol{y})$. An arc $(u, v) \in E$ exists if and only if, for an adjacent label pair $\langle y, y' \rangle$ in the trellis, $u = y \wedge v = \langle y, y' \rangle$ or $u = \langle y, y' \rangle \wedge v = y'$. For the trellis shown in Fig.1, the resulting DAG is shown in Fig.2. Hereafter, it is assumed that G has only one node whose in-degree is zero, denoted by $\mathrm{src}(G)$. It is also assumed that G has only one node whose out-degree is zero, denoted by $\mathrm{snk}(G)$. We abbreviate $\mathrm{src}(G)$ and $\mathrm{snk}(G)$ to src and snk respectively if G is obvious from the context. It is straightforward to extend the following discussion to DAGs with multiple source and sink nodes. Let $\Psi(G)$ be the set of directed paths whose starting node is $\mathrm{src}(G)$ and end node is $\mathrm{snk}(G)$. In this setting, summations over all the sequences in a trellis $\sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})}$ is now rewritten as summations over the set of directed paths $\Psi(G)$, i.e., $\sum_{\pi \in \Psi(G)}$. We can also replace any function defined on $\mathcal{C}(\boldsymbol{y})$, i.e., $f(c)$ $(c \in \mathcal{C}(\boldsymbol{y}))$ with functions defined on V, i.e., $f(v)$ $(v \in V)$. For the sake of simplicity, we treat a directed path π in G as equivalent to the set of nodes appearing on π.

2.1 Standard Formulation

In this subsection, we will formalize an efficient algorithm to calculate summations of the form (3). Note that the summation $\sum_{\boldsymbol{y} \in \mathcal{Y}(G)}$ in the original formulation now becomes $\sum_{\pi \in \Psi(G)}$ in the new formulation. Readers are requested to read them as interchangeable.

Theorem 1 (Generalized Forward Algorithm). *Consider a DAG $G = (V, E)$. Let f_k $(k = 1, \ldots, K)$ be K complex-valued functions defined on V, and ϕ be a complex-valued function defined on V. For a directed path π in G, let $F_k(\pi) := \sum_{v \in \pi} f_k(v)$ $(k = 1, \ldots, K)$, and $\Phi(\pi) := \prod_{v \in \pi} \phi(v)$. Then, for non-negative integers $n_1, \ldots, n_K$,*[1]

[1] In this paper, we define $0^0 := 1$.

$$\sum_{\pi\in\Psi(G)} \Phi(\pi){F_1}^{n_1}(\pi)\cdots{F_K}^{n_K}(\pi) = \alpha_{n_1,\ldots,n_K}(\mathrm{snk}) \ , \tag{4}$$

where α is defined recursively along a topological order of V, as follows.

$$\begin{aligned}
&\alpha_{n_1,\ldots,n_K}(\mathrm{src}) := \phi(\mathrm{src}){f_1}^{n_1}(\mathrm{src})\cdots{f_K}^{n_K}(\mathrm{src}) \ , \\
&\alpha_{n_1,\ldots,n_K}(v) := \phi(v)\sum_{m_1=0}^{n_1}\left[\binom{n_1}{m_1}{f_1}^{m_1}(v)\cdots\sum_{m_K=0}^{n_K}\left[\binom{n_K}{m_K}{f_K}^{m_K}(v)\right.\right. \\
&\qquad\left.\left.\times\sum_{v'\in\mathrm{prev}(v)}\alpha_{n_1-m_1,\ldots,n_K-m_K}(v')\right]\cdots\right] \qquad (v\neq\mathrm{src}) \ .
\end{aligned} \tag{5}$$

Here, $\binom{n}{m}$ is the binomial coefficient, and $\mathrm{prev}(v)$ *denotes the set of preceding nodes of v, i.e.,* $\mathrm{prev}(v) := \{x|(x,v)\in E\}$.

Proof. Hereafter, we will prove the following claim (6) from the recursive definition of α (5).

$$\alpha_{n_1,\ldots,n_K}(v) = \sum_{\pi\in\Psi_{\mathrm{src}\to v}(G)} \Phi(\pi){F_1}^{n_1}(\pi)\cdots{F_K}^{n_K}(\pi) \qquad \left({}^{\forall}v\in V\right) \ , \tag{6}$$

where $\Psi_{u\to v}(G)$ $(u,v\in V)$ is the set of directed paths whose starting node is u and end node is v. We utilize mathematical induction along a topological order of V.

For $v=\mathrm{src}$, (6) is obviously true from the definition of α (5).

Hereafter, for the sake of simplicity, we just refer to the case where there is only one function for F_k, i.e., $K=1$. We also restate F_1 and n_1 as F and n respectively.

For a given $v\in V$, if we assume that (6) is true for ${}^{\forall}v'\in\mathrm{prev}(v)$, then

$$\begin{aligned}
\alpha_n(v) &= \phi(v)\sum_{m=0}^{n}\left[\binom{n}{m}f^m(v)\sum_{v'\in\mathrm{prev}(v)}\alpha_{n-m}(v')\right] \\
&\qquad\text{(from the definition of } \alpha \text{ in (5))} \\
&= \phi(v)\sum_{m=0}^{n}\left[\binom{n}{m}f^m(v)\sum_{v'\in\mathrm{prev}(v)}\sum_{\pi\in\Psi_{\mathrm{src}\to v'}(G)}\Phi(\pi)F^{n-m}(\pi)\right] \\
&\qquad\text{(from the inductive assumption (6) for } {}^{\forall}v'\in\mathrm{prev}(v)) \\
&= \sum_{v'\in\mathrm{prev}(v)}\sum_{\pi\in\Psi_{\mathrm{src}\to v'}(G)}\left[\phi(v)\Phi(\pi)\sum_{m=0}^{n}\binom{n}{m}f^m(v)F^{n-m}(\pi)\right] \\
&= \sum_{v'\in\mathrm{prev}(v)}\sum_{\pi\in\Psi_{\mathrm{src}\to v'}(G)}\phi(v)\Phi(\pi)\left(f(v)+F(\pi)\right)^n \\
&\qquad(\because \text{binomial theorem}) \\
&= \sum_{\pi\in\Psi_{\mathrm{src}\to v}(G)}\Phi(\pi)F^n(\pi) \ .
\end{aligned} \tag{7}$$

So (6) appears to be true for v. Thus, (6) is true for ${}^{\forall}v \in V$, including the case $v = \mathrm{snk}$, which is equivalent to (4) because $\Psi_{\mathrm{src}\to\mathrm{snk}}(G) = \Psi(G)$. We can also provide the proof of (6) for the case where there are multiple indices for α in a similar manner. □

This theorem shows that recursive calculation of α leads to the calculation of the left-hand side of (4). We call this recursive calculation $(n_1, \ldots, n_K)$*-th order forward algorithm*. It is worth noting that the specialization for the case $K = 0$, i.e., the (0)-th order forward algorithm is equivalent to the forward procedure of the ordinary forward-backward algorithm. Of course α_0 is equivalent to the ordinary forward variable (often denoted by α), too.

From the definition of α in (5), it is easy to show that the time complexity is proportional to $(|V| + |E|)\,(n_1+1)^2 \cdots (n_K+1)^2$, assuming that $\binom{n}{m}$ and ${f_k}^{n_k}(v)$ can be calculated in constant time.

In the ordinary forward-backward algorithm, we also calculate backward variables, that are often denoted by β. $\alpha(v)\beta(v)/\phi(v)$ is the summation of $\Phi(\pi)$ over the sequences appearing in the sub-trellis constrained on a node v. In other words, we can regard $\alpha(v)\beta(v)/\phi(v)$ as the marginal of $\Phi(\pi)$ on v. So, for a function $\acute{f}(v)$, summing up $\acute{f}(v)\alpha(v)\beta(v)/\phi(v)$ on all nodes leads to the summation of $\Phi(\pi)\sum_{v\in\pi}\acute{f}(v)$. This formulation is useful in the case where $\acute{f}$ is sparse, i.e., $\acute{f}(v) = 0$ for most of $v \in V$. Likewise, we can define generalized backward variables $\beta_{n_1,\ldots,n_K}$ in a similar manner to the definition of α in Th.1. A generalized forward variable $\alpha_{n_1,\ldots,n_K}(v)$ is equal to the summation of $\Phi(\pi){F_1}^{n_1}(\pi)\cdots{F_K}^{n_K}(\pi)$ in the sub-trellis prior to v (see (6)), and a generalized β is equal to the summation of the same form in the sub-trellis posterior to v. Thus, a proper combination of generalized α and β yields the summation of $\Phi(\pi){F_1}^{n_1}(\pi)\cdots{F_K}^{n_K}(\pi)$ over the sub-trellis constrained on v. In other words, such a combination can be regarded as the marginal of $\Phi(\pi){F_1}^{n_1}(\pi)\cdots{F_K}^{n_K}(\pi)$ on v. So, we can make a similar discussion on a sparse function $\acute{f}(v)$ in our generalization. This argument is formally summarized in the following theorem.

Theorem 2 (Generalized Forward-backward Algorithm). *In addition to the definitions of $G, f_k, F_k, \phi, \Phi, \alpha$ in Th.1, let $\acute{f}$ be a complex-valued function defined on V, and, for a directed path π in G, let $\acute{F}(\pi) := \sum_{v\in\pi}\acute{f}(v)$. Then,*

$$\begin{aligned}
&\sum_{\pi\in\Psi(G)} \Phi(\pi){F_1}^{n_1}(\pi)\cdots{F_K}^{n_K}(\pi)\acute{F}(\pi) \\
&\quad= \sum_{v\in V}\Bigg[\acute{f}(v)\sum_{m_1=0}^{n_1}\Bigg[\binom{n_1}{m_1}\cdots\sum_{m_K=0}^{n_K}\Bigg[\binom{n_K}{m_K} \\
&\qquad\qquad\times \alpha_{m_1,\ldots,m_K}(v)\beta_{n_1-m_1,\ldots,n_K-m_K}(v)\Bigg]\cdots\Bigg]\Bigg] ,
\end{aligned} \tag{8}$$

where β is recursively defined along a reversed topological order of V, as follows.

$$\beta_{n_1,\ldots,n_K}(\mathrm{snk}) := \begin{cases} 1 & (n_1,\cdots,n_K = 0) \ , \\ 0 & \text{otherwise} \ , \end{cases}$$

$$\begin{aligned}
&\beta_{n_1,\ldots,n_K}(v) \\
&:= \sum_{v'\in \mathrm{next}(v)} \left[\phi(v') \sum_{m_1=0}^{n_1} \left[\binom{n_1}{m_1} {f_1}^{m_1}(v') \cdots \right.\right. \\
&\qquad \left.\left. \times \sum_{m_K=0}^{n_K} \left[\binom{n_K}{m_K} {f_K}^{m_K}(v') \beta_{n_1-m_1,\cdots,n_K-m_K}(v') \right] \cdots \right] \right] \quad (v \neq \mathrm{snk}) \quad ,
\end{aligned} \tag{9}$$

where $\mathrm{next}(v)$ *denotes the set of succeeding nodes of* v*, i.e.,* $\mathrm{next}(v) := \{x | (v,x) \in E\}$.

Some readers may be confused by the definition of β in (9) because the definitions of α (in (5)) and β are asymmetric by design. In the ordinary forward-backward algorithm, α and β are usually defined in a symmetric manner, and combined in the product $\alpha(v)\beta(v)/\phi(v)$ because both $\alpha(v)$ and $\beta(v)$ have the same factor $\phi(v)$ redundantly. However, such a symmetric definition makes the combination of α and β in (8) quite complex. So we define β as in (9).

With the following relation

$$\sum_{\pi\in\Psi(G)} \Gamma(\pi)\acute{F}(\pi) = \sum_{v\in V} \left[\acute{f}(v) \sum_{\pi\in\Psi_v(G)} \Gamma(\pi) \right] \quad , \tag{10}$$

where Γ is a complex-valued function defined on $\Psi(G)$ and $\Psi_v(G) := \{\pi \mid \pi \in \Psi(G), v \in \pi\}$, we give proof of Th.2.

Proof (of Th.2). Just like the proof of Th.1, we can prove the following claim.

$$\beta_{n_1,\ldots,n_K}(v) = \sum_{\pi\in\Psi_{v\leftarrow\mathrm{snk}}(G)} \Phi(G) {F_1}^{n_1}(\pi) \cdots {F_K}^{n_K}(\pi) \quad \left({}^{\forall}v \in V, v \neq \mathrm{snk}\right) \quad , \tag{11}$$

where $\Psi_{v\leftarrow\mathrm{snk}}(G) := \bigcup_{v'\in\mathrm{next}(v)} \Psi_{v'\rightarrow\mathrm{snk}}(G)$. Hereafter, for the sake of simplicity, we just refer to the case where $K = 1$. Obviously, we can split a directed path containing v into two parts. One is prior to v, and the other is posterior to v. In other words, for ${}^{\forall}v \in V$, $\Psi_v(G)$ is the direct product of $\Psi_{\mathrm{src}\rightarrow v}(G)$ and $\Psi_{v\leftarrow\mathrm{snk}}(G)$. So,

$$\begin{aligned}
&\sum_{\pi\in\Psi(G)} \Phi(\pi) F^n(\pi) \acute{F}(\pi) \\
&\quad = \sum_{v\in V} \left[\acute{f}(v) \sum_{\pi\in\Psi_v(G)} \Phi(\pi) F^n(\pi) \right] \\
&\qquad (\because (10) \text{ with } \Gamma(\pi) = \Phi(\pi) F^n(\pi)) \\
&\quad = \sum_{v\in V} \left[\acute{f}(v) \sum_{\pi\in\Psi_{\mathrm{src}\rightarrow v}(G)} \sum_{\pi'\in\Psi_{v\leftarrow\mathrm{snk}}(G)} \Phi(\pi\cup\pi') F^n(\pi\cup\pi') \right] \\
&\quad = \sum_{v\in V} \left[\acute{f}(v) \sum_{\pi\in\Psi_{\mathrm{src}\rightarrow v}(G)} \sum_{\pi'\in\Psi_{v\leftarrow\mathrm{snk}}(G)} \Phi(\pi)\Phi(\pi') \left(F(\pi) + F(\pi')\right)^n \right]
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{v\in V}\left[\acute{f}(v)\sum_{\pi\in\Psi_{\mathrm{src}\to v}(G)}\sum_{\pi'\in\Psi_{v\leftarrow\mathrm{snk}}(G)}\left[\Phi(\pi)\Phi(\pi')\right.\right.\\
&\qquad\left.\left.\times\sum_{m=0}^{n}\binom{n}{m}F^m(\pi)F^{n-m}(\pi')\right]\right]\\
&\qquad(\because \text{binomial theorem})\\
&= \sum_{v\in V}\left[\acute{f}(v)\sum_{m=0}^{n}\binom{n}{m}\left(\sum_{\pi\in\Psi_{\mathrm{src}\to v}(G)}\Phi(\pi)F^m(\pi)\right)\right.\\
&\qquad\left.\times\left(\sum_{\pi'\in\Psi_{v\leftarrow\mathrm{snk}}(G)}\Phi(\pi')F^{n-m}(\pi')\right)\right] \qquad (12)\\
&= \sum_{v\in V}\left[\acute{f}(v)\sum_{m=0}^{n}\binom{n}{m}\alpha_m(v)\beta_{n-m}(v)\right]\\
&\qquad(\because (6),\ (11))\ .
\end{aligned}
$$

□

This theorem shows that, by recursive calculation of α and β, and storing them for $^\forall v \in V$, we can calculate the summation appearing in the left-hand side of (8). We call this calculation $(n_1, \ldots, n_K)$*-th order forward-backward algorithm*. It is worth noting that the (0)-th order forward-backward algorithm is equivalent to the ordinary forward-backward algorithm for the form of (2).

In the same way as Th.1, the time complexity is proportional to $(|V| + |E|)\,(n_1 + 1)^2 \cdots (n_K + 1)^2$. The space complexity is proportional to $|V|(n_1 + 1) \cdots (n_K + 1)$ because we need to store α and β for $^\forall v \in V$.

2.2 Fourier-Transformed Formulation

In both Th.1 and Th.2, we can find that the definitions of α and β involve a sort of convolution. It is well known that a convolution is transformed into element-wise product under the Fourier transform. So, we can derive formulation with element-wise products instead of convolutions under the Fourier transform.

First, let us give a brief description of convolution and its relationship with the Fourier transform. Consider two complex-valued sequences $\{a_n\}, \{b_n\}$ $(n = 0, \ldots, 2N-1)$. Both have N trailing zeros in them, i.e., $a_n = 0, b_n = 0$ $(n = N, \ldots, 2N-1)$. Let a sequence $\{c_n\}$ be as follows.

$$
c_n := \sum_{m=0}^{n} a_m b_{n-m} \qquad (n = 0, \ldots, 2N-1)\ . \tag{13}
$$

Then,

$$
\mathcal{F}(\{c_n\}) = \mathcal{F}(\{a_n\}) \circ \mathcal{F}(\{b_n\})\ , \tag{14}
$$

where $\mathcal{F} : \mathbb{C}^{2N} \to \mathbb{C}^{2N}$ is the discrete Fourier transform [2], and $\circ$ denotes an element-wise product. The relation (14) is often referred to as the convolution theorem.

Convolutions in the definitions of α and β are of the form

$$c_n := \sum_{m=0}^{n} \binom{n}{m} a_m b_{n-m} \quad , \tag{15}$$

which is slightly different from the ordinary convolution in (13). However, with the definition of the binomial coefficient, we can rephrase (15) as

$$\frac{c_n}{n!} = \sum_{m=0}^{n} \frac{a_m}{m!} \cdot \frac{b_{n-m}}{(n-m)!} \quad , \tag{16}$$

in which the sequence $\left\{\frac{c_n}{n!}\right\}$ is calculated by an ordinary convolution of $\left\{\frac{a_n}{n!}\right\}$ and $\left\{\frac{b_n}{n!}\right\}$.

Based on the convolution theorem mentioned above, we will derive the Fourier-transformed versions of Th.1 and Th.2. For the sake of simplicity, we just refer to the case where $K = 1$.

Theorem 3 (Fourier-transformed Generalized Forward Algorithm). *In addition to the definitions of $\phi, f := f_1, \alpha$ in Th.1, let*

$$\left\{\tilde{f}_n(v)\right\} := \mathcal{F}\left(\left\{\frac{f^0(v)}{0!}, \ldots, \frac{f^{N-1}(v)}{(N-1)!}, 0, \ldots, 0\right\}\right) \qquad \left(^\forall v \in V\right) \quad , \tag{17}$$

and

$$\begin{aligned} &\tilde{\alpha}_n(\mathrm{src}) := \phi(\mathrm{src})\tilde{f}_n(\mathrm{src}) \quad , \\ &\tilde{\alpha}_n(v) := \phi(v)\tilde{f}_n(v) \sum_{v' \in \mathrm{prev}(v)} \tilde{\alpha}_n(v') \qquad (v \neq \mathrm{src}) \quad . \end{aligned} \tag{18}$$

Then, for $0 \leqq n \leqq N-1$ and $^\forall v \in V$,

$$\frac{\alpha_n(v)}{n!} = \left\{\mathcal{F}^{-1}\left(\{\tilde{\alpha}_n(v)\}\right)\right\}_n \quad . \tag{19}$$

Proof. (Sketch) From the Fourier transform for the definition of α in (5), linearity of the Fourier transform and the convolution theorem (14), we can easily show the proof. □

Theorem 4 (Fourier-transformed Generalized Forward-backward Algorithm). *In addition to the definitions of $\phi, f, \tilde{f}, \tilde{\alpha}$ in Th.3, let*

$$\Phi(\pi) := \prod_{v \in \pi} \phi(v), \quad F(\pi) := \sum_{v \in \pi} f(v), \quad \acute{F}(\pi) := \sum_{v \in \pi} \acute{f}(v) \tag{20}$$

for a directed path π in G, and

[2] We use normalization constant 1 for $\mathcal{F}$, and $\frac{1}{2N}$ for the inverse discrete Fourier transform $\mathcal{F}^{-1}$.

$$\begin{aligned}
&\tilde{\beta}_n(\mathrm{snk}) := 1 \ , \\
&\tilde{\beta}_n(v) := \sum_{v' \in \mathrm{next}(v)} \phi(v') \tilde{f}_n(v') \tilde{\beta}_n(v') \ , \\
&\tilde{S}_n := \sum_{v \in V} \acute{f}(v) \tilde{\alpha}_n(v) \tilde{\beta}_n(v) \ .
\end{aligned} \tag{21}$$

Then, for $0 \leqq n \leqq N-1$,

$$\frac{1}{n!} \sum_{\pi \in \Psi(G)} \Phi(\pi) F^n(\pi) \acute{F}(\pi) = \left\{ \mathcal{F}^{-1} \left(\left\{ \tilde{S}_n \right\} \right) \right\}_n \ . \tag{22}$$

Proof. (Sketch) The same as the proof of Th.3, with the Fourier transform of (8). □

The argument above can be extended to the case where there are multiple indices. We can apply the Fourier transform only to an arbitrary subset of indices, and we may leave the remaining indices as they are.

Using the fast Fourier transform (FFT), the proportionality factor of an index n_k in the time complexity is reduced from $(n_k+1)^2$ to $(n_k+1)\log(n_k+1)$.

3 Application of the Proposed Algorithm

In this section, we will show that some specializations of the generalization derived above correspond to some calculations studied in past research. Our intuition behind the following derivations is simple. Problems all boil down to how we can transform the summation at hand into an expression of the form (3). Once we obtain a formula of the form (3), our theorems immediately offer an efficient algorithm for calculating it.

In the following subsections, we assume that the distribution over output sequences is modeled by CRFs, which is defined as follows.

$$P(\pi|\boldsymbol{x};\boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{\lambda};\boldsymbol{x})} \sum_{\pi \in \Psi(G)} \left(\prod_{v \in \pi} \phi(\boldsymbol{x}, v) \right) \tag{23}$$

with

$$Z(\boldsymbol{\lambda};\boldsymbol{x}) := \sum_{\pi \in \Psi(G)} \left(\prod_{v \in \pi} \phi(\boldsymbol{x}, v) \right) \ , \tag{24}$$

$$\phi(\boldsymbol{x}, v) := \exp\left(\sum_k \lambda_k f_k(\boldsymbol{x}, v) \right) \ . \tag{25}$$

We also assume that all the expectations and covariances are taken under (23).

3.1 Covariance for Conditional Random Fields

For CRFs, it is sometimes necessary to calculate the covariance of feature functions. For feature functions $F_1(\pi), F_2(\pi)$ $(\pi \in \Psi(G))$, the covariance is

$$\mathrm{Cov}\left[F_1(\pi), F_2(\pi)\right] = E\left[F_1(\pi)F_2(\pi)\right] - E\left[F_1(\pi)\right] E\left[F_2(\pi)\right] \ . \tag{26}$$

If both F_1 and F_2 are of the form $\sum_{v\in\pi} f(v)$, the second term of (26) can be calculated either by the (1)-st order forward algorithm or the (0)-th order forward-backward algorithm, and the first term of (26) can be calculated either by the $(1,1)$-th order forward algorithm or the (1)-st order forward-backward algorithm.

Covariance of feature functions is useful for optimization of CRFs. Because it has rich information of curvature of the log-likelihood objective function, i.e.,

$$\frac{\partial^2}{\partial\lambda_i \partial\lambda_j}\mathcal{L}(\boldsymbol{\lambda}) = -\mathrm{Cov}[F_i(\boldsymbol{x},\pi), F_j(\boldsymbol{x},\pi)] \ , \tag{27}$$

where $\mathcal{L}$ is the log-likelihood objective function and $F_k(\boldsymbol{x},\pi) := \sum_{v\in\pi} f_k(\boldsymbol{x}, v)$. For example, S. V. N. Vishwanathan et al. [3] used stochastic gradient methods to accelerate the optimization of the log-likelihood objective for CRFs. In stochastic gradient methods, it is required to calculate the Hessian-vector product. Although they use automatic differentiation to calculate the Hessian-vector product, we can derive an alternative algorithm for it as a specialization of the generalized forward-backward algorithm. For a vector $\boldsymbol{v}$, Hessian-vector product is

$$\begin{aligned} \{\boldsymbol{H}(\mathcal{L})\cdot\boldsymbol{v}\}_i &= -\sum_j \mathrm{Cov}[F_i(\boldsymbol{x},\pi), F_j(\boldsymbol{x},\pi)]v_j \\ &= E[F_i(\boldsymbol{x},\pi)]E[\boldsymbol{F}(\boldsymbol{x},\pi)\cdot\boldsymbol{v}] - E[F_i(\boldsymbol{x},\pi)\left(\boldsymbol{F}(\boldsymbol{x},\pi)\cdot\boldsymbol{v}\right)] \ , \end{aligned} \tag{28}$$

where $\boldsymbol{H}$ is the Hessian matrix. The (1)-st order forward-backward algorithm provides an efficient algorithm to calculate the second term in (28). That algorithm is equivalent to the calculation by the automatic differentiation as far as the computational cost is concerned. [3]

As another practical application of the covariance, we can take differentiation of more complicated objective functions than the log-likelihood objective. For example, the objective function used in Jiao et al. [4] has the conditional entropy of CRFs.

$$\mathcal{H}(\boldsymbol{\lambda}) = E[\log P(\pi|\boldsymbol{x};\boldsymbol{\lambda})] = E[\boldsymbol{F}(\boldsymbol{x},\pi)\cdot\boldsymbol{\lambda}] - \log Z(\boldsymbol{\lambda};\boldsymbol{x}) \ . \tag{29}$$

[3] Generally speaking, the algorithm implicitly derived by the automatic differentiation for the ordinary forward-backward algorithm is equivalent to the (1)-st order forward-backward algorithm. In addition, we conjecture that the higher-order automatic differentiation for the ordinary forward-backward algorithm is essentially equivalent to the generalized forward-backward algorithm proposed in this paper.

The ordinary ((0)-th order) forward-backward algorithm can calculate the gradient of the second term. The gradient of the first term is

$$\frac{\partial}{\partial \lambda_k} E[\boldsymbol{F}(\boldsymbol{x}, \pi) \cdot \boldsymbol{\lambda}] = \mathrm{Cov}\left[\boldsymbol{F}(\boldsymbol{x}, \pi) \cdot \boldsymbol{\lambda}, F_k(\boldsymbol{x}, \pi)\right] \ . \tag{30}$$

The discussion of the calculation of (30) is the same as the one for the second term of (28).

In more general formulation, for the expectation of a sufficient statistic $\bar{F}(\pi)$ ($\pi \in \Psi(G)$),

$$\frac{\partial}{\partial \lambda_k} E[\bar{F}(\pi)] = \mathrm{Cov}\left[\bar{F}(\pi), F_k(\boldsymbol{x}, \pi)\right] \ . \tag{31}$$

(31) falls within the applicable scope of the generalization proposed in this paper, assuming $\bar{F}$ is of the form $\bar{F}(\pi) = \sum_{v \in \pi} \bar{f}(v)$.

3.2 Hamming Loss for Conditional Random Fields

Kakade et al. [5] proposed to use Hamming loss objective function in parameter estimations involved in sequential labeling. For a correctly annotated input-output sequence pair $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$, the objective to be optimized is

$$C_1(\boldsymbol{\lambda}; \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) := \frac{1}{T} \sum_t \log P(\hat{y}_t | \hat{\boldsymbol{x}}; \boldsymbol{\lambda}) \ . \tag{32}$$

With notations used in this paper, we can rephrase this objective as follows.

$$C_1(\boldsymbol{\lambda}; \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = \frac{1}{T} \sum_t \log \sum_{\pi \in \Psi(G)} P(\pi | \hat{\boldsymbol{x}}; \boldsymbol{\lambda}) \delta_{\hat{\pi}_t}(\pi) \ , \tag{33}$$

where $\hat{\pi}_t$ is the node that corresponds to $\hat{y}_t$. T is the number of labels in $\hat{\boldsymbol{y}}$, and $\delta_{\hat{\pi}_t}(\pi) := \sum_{v \in \pi} \delta_{v, \hat{\pi}_t}$ is the indicator for the condition $\hat{\pi}_t \in \pi$. Its gradient is

$$\begin{aligned} \frac{\partial C_1}{\partial \lambda_k} = \frac{1}{T} \sum_t & \frac{\sum_{\pi \in \Psi(G)} P(\pi | \boldsymbol{x}; \boldsymbol{\lambda}) F_k(\boldsymbol{x}, \pi) \delta_{\hat{\pi}_t}(\pi)}{\sum_{\pi \in \Psi(G)} P(\pi | \boldsymbol{x}; \boldsymbol{\lambda}) \delta_{\hat{\pi}_t}(\pi)} \\ & - \sum_{\pi \in \Psi(G)} P(\pi | \boldsymbol{x}; \boldsymbol{\lambda}) F_k(\boldsymbol{x}, \pi) \ . \end{aligned} \tag{34}$$

The second term is of the form which can be calculated by the ordinary ((0)-th order) forward-backward algorithm.

To calculate the first term in (34), Kakade et al. [5] derived a sort of the forward-backward algorithm, which is as efficient as the ordinary forward-backward algorithm. We can show that a specialization of the generalization proposed in this paper also leads to an equivalent algorithm.

With following definitions

$$c(v) := \begin{cases} 1 & (v \text{ represents a correct label}) \ , \\ 0 & (\text{otherwise}) \ , \end{cases} \tag{35}$$

$$\alpha_0(v) := \begin{cases} \phi(v) & (v = \mathrm{src}) \ , \\ \phi(v) \sum_{v' \in \mathrm{prev}(v)} \alpha_0(v') & (\mathrm{otherwise}) \ , \end{cases} \tag{36}$$

$$\alpha_1(v) := \begin{cases} \phi(v) \frac{c(v)}{\alpha_0(v)\beta_0(v)} & (v = \mathrm{src}) \ , \\ \frac{c(v)}{\alpha_0(v)\beta_0(v)} \alpha_0(v) + \phi(v) \sum_{v' \in \mathrm{prev}(v)} \alpha_1(v') & (\mathrm{otherwise}) \ , \end{cases} \tag{37}$$

$$\beta_0(v) := \begin{cases} 1 & (v = \mathrm{snk}) \ , \\ \sum_{v' \in \mathrm{next}(v)} \phi(v')\beta_0(v') & (\mathrm{otherwise}) \ , \end{cases} \tag{38}$$

$$\beta_1(v) := \begin{cases} 0 & (v = \mathrm{snk}) \ , \\ \frac{c(v)}{\alpha_0(v)\beta_0(v)} \beta_0(v) + \sum_{v' \in \mathrm{next}(v)} \phi(v')\beta_1(v') & (\mathrm{otherwise}) \ , \end{cases} \tag{39}$$

and noting the following relation

$$\begin{aligned} \Delta(\pi) &:= \sum_t \frac{\delta_{\hat{\pi}_t}(\pi)}{\alpha_0(\hat{\pi}_t)\beta_0(\hat{\pi}_t)} = \sum_t \frac{\sum_{v \in \pi} \delta_{v,\hat{\pi}_t}}{\alpha_0(\hat{\pi}_t)\beta_0(\hat{\pi}_t)} \\ &= \sum_{v \in \pi} \left(\sum_t \frac{\delta_{v,\hat{\pi}_t}}{\alpha_0(\hat{\pi}_t)\beta_0(\hat{\pi}_t)} \right) = \sum_{v \in \pi} \frac{c(v)}{\alpha_0(v)\beta_0(v)} \ , \end{aligned} \tag{40}$$

we can derive an efficient algorithm to calculate the first term in (34) as follows.

$$\begin{aligned} &\frac{1}{T} \sum_t \frac{\sum_{\pi \in \Psi(G)} P(\pi|\boldsymbol{x};\boldsymbol{\lambda}) F_k(\boldsymbol{x},\pi) \delta_{\hat{\pi}_t}(\pi)}{\sum_{\pi \in \Psi(G)} P(\pi|\boldsymbol{x};\boldsymbol{\lambda}) \delta_{\hat{\pi}_t}(\pi)} \\ &\quad = \frac{1}{T} \sum_t \frac{\sum_{\pi \in \Psi(G)} P(\pi|\boldsymbol{x};\boldsymbol{\lambda}) F_k(\boldsymbol{x},\pi) \delta_{\hat{\pi}_t}(\pi)}{\alpha_0(\hat{\pi}_t)\beta_0(\hat{\pi}_t)} \\ &\qquad (\because \text{(0)-th order forward-backward algorithm for the denominator}) \\ &\quad = \frac{1}{T} \sum_{\pi \in \Psi(G)} \left(P(\pi|\boldsymbol{x};\boldsymbol{\lambda}) F_k(\boldsymbol{x},\pi) \sum_t \frac{\delta_{\hat{\pi}_t}(\pi)}{\alpha_0(\hat{\pi}_t)\beta_0(\hat{\pi}_t)} \right) \\ &\quad = \frac{1}{T} \sum_{\pi \in \Psi(G)} P(\pi|\boldsymbol{x};\boldsymbol{\lambda}) F_k(\boldsymbol{x},\pi) \Delta(\pi) \\ &\quad = \frac{1}{T} \sum_{v \in V} f_k(\boldsymbol{x},v) \left(\alpha_1(v)\beta_0(v) + \alpha_0(v)\beta_1(v) \right) \\ &\qquad (\because \text{(1)-st order forward-backward algorithm}) \ . \end{aligned} \tag{41}$$

Note that $\alpha_1(v)$ and $\beta_1(v)$ in this notation respectively correspond to $\omega_{i+1}^{\mathrm{f}}$ and ω_i^{b} used in [5]. Of course both sides are also equivalent concerning the computational cost.

3.3 Generalized Expectation Criteria for Conditional Random Fields

Mann et al. [6] proposed a semi-supervised training method called generalized expectation criteria to improve accuracy with unlabeled data. The objective used in [6] has an additional term

$$-\theta D(\hat{P}||\tilde{P}) \ , \tag{42}$$

where θ is a given hyper-parameter, D is the KL divergence, $\hat{P}$ is a given target distribution and $\tilde{P}$ is the conditional distribution of labels given a feature $f_m(\boldsymbol{x}, t)$ at time t, i.e., [4]

$$\tilde{P} := \tilde{P}(y_t | f_m(\boldsymbol{x}, t) = 1; \boldsymbol{\lambda}) \ . \tag{43}$$

We now focus on the parameter gradient. Instead of the derivation shown in [6], we first rephrase $\tilde{P}(y|f_m(x,t) = 1; \boldsymbol{\lambda})$ as

$$\tilde{P}(\ell(v)|f_m(v) = 1; \boldsymbol{\lambda}) = \frac{1}{U_m} \sum_{\boldsymbol{x} \in \boldsymbol{U}_m} \sum_{\pi \in \Psi(G)} P(\pi|\boldsymbol{x}; \boldsymbol{\lambda}) \Delta_{f_m, y}(\pi) \ , \tag{44}$$

where $\ell(v)$ is the label assigned to v, $\boldsymbol{U}_m$ is the set of sequences where f_m is present for some nodes, and

$$\Delta_{f_m, y}(\pi) = \sum_{v \in \pi} f_m(v) \delta_{\ell(v), y} \ . \tag{45}$$

Noting the following relation

$$\Delta_{f_m}(\pi) := \sum_l \frac{\hat{P}}{\tilde{P}} \Delta_{f_m, l}(\pi) = \sum_{v \in \pi} \frac{\hat{P}(\ell(v)|f_m(v) = 1)}{\tilde{P}(\ell(v)|f_m(v) = 1; \boldsymbol{\lambda})} f_m(v) \tag{46}$$

we get

$$\frac{\partial}{\partial \lambda_k} D(\hat{P}||\tilde{P}) = -\frac{1}{U_m} \sum_{\boldsymbol{x} \in \boldsymbol{U}_m} \sum_l \frac{\hat{P}}{\tilde{P}} \frac{\partial}{\partial \lambda_k} E\left[\Delta_{f_m, l}(\pi)\right] \ , \tag{47}$$

and

$$\sum_l \frac{\hat{P}}{\tilde{P}} \frac{\partial}{\partial \lambda_k} E\left[\Delta_{f_m, l}(\pi)\right] = -E[\Delta_{f_m}(\pi) F_k(\boldsymbol{x}, \pi)] + E[\Delta_{f_m}(\pi)] E[F_k(\boldsymbol{x}, \pi)] \ . \tag{48}$$

Here, the first term can be calculated by the (1)-st order forward-backward algorithm, the second term by the ordinary ((0)-th order) forward-backward algorithm.

Note that this calculation can drastically reduce the computational cost compared with the algorithm proposed in [6]. Computational cost needed by our formulation is scaled by up to a constant factor compared with the ordinary forward-backward algorithm, whereas [6] takes the forward-backward algorithm for each label so computational cost is proportional to the number of labels.

[4] In the same way as their paper, we assume that all output sequences are same in length and f_m is binary.

4 Further Possibilities

In the previous section, we show some specializations of our generalized forward-backward algorithm. However, we have not fully utilized the potential capacity of the higher order generalization of the forward-backward algorithm we proposed in this paper. To show a sample application of the higher order generalization, we will describe the expected F-measure optimization.

4.1 Expected F-Measure Optimization

Jansche [7] utilized the expected f-measure as an objective function in optimization of logistic regression models. He has just described on single label, non-structured case. Here, let this criteria be sequentially-extended. We define

$$\mathcal{F}_\gamma(\boldsymbol{\lambda}; \hat{\boldsymbol{x}}) := E[\mathcal{F}_\gamma(\pi)] \ , \tag{49}$$

where the expectation is taken under (23), and $\mathcal{F}_\gamma(\pi)$ is the label-wise f-measure for a given output sequence π, i.e.,

$$\mathcal{F}_\gamma(\pi) := \frac{(1+\gamma^2)l(\pi)}{|\pi| + \gamma^2 L} \ . \tag{50}$$

L is the number of labels in the correctly annotated answer sequence, and $l(\pi)$ is the number of labels in the output sequence π that coincide with the correct labels. $|\pi|$ is the total number of labels appearing in π. Hereafter, we refer only to the case $\gamma = 1$. Let $\mathcal{F}(\pi) := \mathcal{F}_1(\pi)$. For the case where all the possible output sequences have the same number of labels (length), optimization concerning (49) – calculations of value and gradient – requires up to the $(1,1)$-th order forward-backward algorithm. On the other hand, for the case where the lengths of the output sequences are different as in [8], because the denominator in (49) is variable in the summation, calculations of the value and the gradient of (49) are not trivial at all.

However, with the Taylor expansion of the reciprocal function in (49), we get

$$\begin{aligned} E[\mathcal{F}(\pi)] &= \frac{1}{Z(\boldsymbol{\lambda}; \boldsymbol{x})} \sum_{\pi \in \Psi(G)} \Phi(\pi) \frac{2l(\pi)}{|\pi| + L} \\ &= \frac{2}{Z(\boldsymbol{\lambda}; \boldsymbol{x})(L + x_0)} \sum_{\pi \in \Psi(G)} \left[\Phi(\pi) l(\pi) \sum_{m=0}^{\infty} \left(\frac{|\pi| - x_0}{-L - x_0} \right)^m \right] \\ &= \frac{2}{Z(\boldsymbol{\lambda}; \boldsymbol{x})(L + x_0)} \sum_{m=0}^{\infty} \sum_{\pi \in \Psi(G)} \Phi(\pi) l(\pi) \left(\frac{|\pi| - x_0}{-L - x_0} \right)^m \ , \end{aligned} \tag{51}$$

where x_0 is a complex value that satisfies the condition $\left| \frac{|\pi| - x_0}{-L - x_0} \right| < 1 \ \ (^\forall \pi \in \Psi(G))$ (such a value really exists at any time). By truncating the higher terms in the Taylor series, we can approximate the value.

$$E[\mathcal{F}(\pi)] \approx \frac{2}{Z(\boldsymbol{\lambda}; \boldsymbol{x})(L + x_0)} \sum_{m=0}^{M} \sum_{\pi \in \Psi(G)} \Phi(\pi) l(\pi) \left(\frac{|\pi| - x_0}{-L - x_0} \right)^m \ . \tag{52}$$

The summation $\sum_{\pi \in \Psi(G)}$ in (52) is of the form to which the $(m, 1)$-th order forward algorithm is applicable. We can also derive an approximation of the gradient of (52).

$$\frac{\partial}{\partial \lambda_k} E[\mathcal{F}(\pi)] = E[\mathcal{F}(\pi) F_k(\boldsymbol{x}, \pi)] - E[\mathcal{F}(\pi)] E[F_k(\boldsymbol{x}, \pi)] \tag{53}$$

The first term in (53) can be approximated by the $(m, 1)$-th forward-backward algorithm in a similar manner to (52). Therefore, we can optimize the objective with numerical optimization routines.

For a given constant M, these calculations are as efficient as the ordinary forward-backward algorithm, scaled by up to a constant factor. For trellises in which the length of a node is up to 5, we have experimentally confirmed that $M \sim 15$ proves to be sufficient to approximate the objective and the gradient with enough precision under the double-precision floating-point arithmetic.

5 Conclusion

In this paper, we proposed a generalization of the forward-backward algorithm, which is applicable to much broader types of summations over all possible sequences than the ordinary forward-backward algorithm. We also show that our theorems in this paper can offer efficient algorithms for some calculations required in past studies, even for the summation of a non-linear function using a Taylor expansion.

We are now investigating other possibilities of applications of our generalization, including optimization of much more complex probabilistic models than log-linear models on sequential labeling tasks (we assume CRFs with polynomial kernels [9] [10], for example).

While we proposed the generalization only for linear chains, it is straightforward to extend this generalization to trees. However, it will be much more challenging to find a counterpart in the case of loopy structures. We are going to study such a possibility.

References

1. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. J. Roy. Stat. Soc. Ser. B 39(1), 1–38 (1977)
2. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. of ICML 2001, pp. 282–289 (2001)
3. Vishwanathan, S.V.N., Schraudolph, N.N., Schmidt, M.W., Murphy, K.P.: Accelerated training of conditional random fields with stochastic gradient methods. In: Proc. of ICML 2006, pp. 969–976 (2006)
4. Jiao, F., Wang, S., Lee, C.-H., Greiner, R., Schuurmans, D.: Semi-supervised conditional random fields for improved sequence segmentation and labeling. In: Proc. of COLING-ACL 2006, July 2006, pp. 209–216 (2006)

5. Kakade, S., Teh, Y.W., Roweis, S.: An alternate objective function for Markovian fields. In: Proc. of the ICML 2002, vol. 19 (2002)
6. Mann, G.S., McCallum, A.: Generalized expectation criteria for semi-supervised learning of conditional random fields. In: Proc. of ACL 2008: HLT, June 2008, pp. 870–878 (2008)
7. Jansche, M.: Maximum expected f-measure training of logistic regression models. In: Proc. of HLT/EMNLP 2005, October 2005, pp. 692–699 (2005)
8. Sarawagi, S., Cohen, W.W.: Semi-markov conditional random fields for information extraction. In: NIPS, vol. 17, pp. 1185–1192 (2004)
9. Altun, Y., Smola, A.J., Hofmann, T.: Exponential families for conditional random fields. In: Proc. of UAI 2004, pp. 2–9 (2004)
10. Lafferty, J., Zhu, X., Liu, Y.: Kernel conditional random fields: Representation and clique selection. In: Proc. of ICML 2004 (2004)