Training Conditional Random Fields with Multivariate Evaluation Measures.

Conference Paper · January 2006

DOI: 10.3115/1220175.1220203 · Source: DBLP

3 authors, including:

Erik McDermott
Google Inc.
74 PUBLICATIONS **1,437** CITATIONS
SEE PROFILE

Hideki Isozaki
Okayama Prefectural University
86 PUBLICATIONS **1,519** CITATIONS
SEE PROFILE

Training Conditional Random Fields with Multivariate Evaluation Measures

Jun Suzuki, Erik McDermott and Hideki Isozaki
NTT Communication Science Laboratories, NTT Corp.
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
{jun, mcd, isozaki}@cslab.kecl.ntt.co.jp

Abstract

This paper proposes a framework for training Conditional Random Fields (CRFs) to optimize multivariate evaluation measures, including non-linear measures such as F-score. Our proposed framework is derived from an error minimization approach that provides a simple solution for directly optimizing *any* evaluation measure. Specifically focusing on sequential segmentation tasks, i.e. text chunking and named entity recognition, we introduce a loss function that closely reflects the target evaluation measure for these tasks, namely, segmentation F-score. Our experiments show that our method performs better than standard CRF training.

1 Introduction

Conditional random fields (CRFs) are a recently introduced formalism (Lafferty et al., 2001) for representing a conditional model $p(\boldsymbol{y}|\boldsymbol{x})$, where both a set of inputs, $\boldsymbol{x}$, and a set of outputs, $\boldsymbol{y}$, display non-trivial interdependency. CRFs are basically defined as a discriminative model of Markov random fields conditioned on inputs (observations) $\boldsymbol{x}$. Unlike generative models, CRFs model only the output $\boldsymbol{y}$'s distribution over $\boldsymbol{x}$. This allows CRFs to use flexible features such as complicated functions of multiple observations. The modeling power of CRFs has been of great benefit in several applications, such as shallow parsing (Sha and Pereira, 2003) and information extraction (McCallum and Li, 2003).

Since the introduction of CRFs, intensive research has been undertaken to boost their effectiveness. The first approach to estimating CRF parameters is the *maximum likelihood (ML)* criterion over conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ itself (Lafferty et al., 2001). The ML criterion, however, is prone to over-fitting the training data, especially since CRFs are often trained with a very large number of correlated features. The *maximum a posteriori (MAP)* criterion over parameters, $\boldsymbol{\lambda}$, given $\boldsymbol{x}$ and $\boldsymbol{y}$ is the natural choice for reducing over-fitting (Sha and Pereira, 2003). Moreover, the Bayes approach, which optimizes both MAP and the prior distribution of the parameters, has also been proposed (Qi et al., 2005). Furthermore, large margin criteria have been employed to optimize the model parameters (Taskar et al., 2004; Tsochantaridis et al., 2005).

These training criteria have yielded excellent results for various tasks. However, real world tasks are evaluated by task-specific evaluation measures, including non-linear measures such as F-score, while all of the above criteria achieve optimization based on the linear combination of average accuracies, or error rates, rather than a given task-specific evaluation measure. For example, *sequential segmentation tasks (SSTs)*, such as text chunking and named entity recognition, are generally evaluated with the *segmentation F-score*. This inconsistency between the objective function during training and the task evaluation measure might produce a suboptimal result.

In fact, to overcome this inconsistency, an SVM-based multivariate optimization method has recently been proposed (Joachims, 2005). Moreover, an F-score optimization method for logistic regression has also been proposed (Jansche, 2005). In the same spirit as the above studies, we first propose a generalization framework for CRF training that allows us to optimize directly not only the error rate, but also any evaluation measure. In other words, our framework can incorporate any evaluation measure of interest into the loss function and then optimize this loss function as the training objective function. Our proposed framework is fundamentally derived from an approach to (smoothed) error rate minimization well

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 217–224,
Sydney, July 2006.

known in the speech and pattern recognition community, namely the *Minimum Classification Error (MCE)* framework (Juang and Katagiri, 1992). The framework of MCE criterion training supports the theoretical background of our method. The approach proposed here subsumes the conventional ML/MAP criteria training of CRFs, as described in the following.

After describing the new framework, as an example of optimizing multivariate evaluation measures, we focus on SSTs and introduce a segmentation F-score loss function for CRFs.

2 CRFs and Training Criteria

Given an input (observation) $\boldsymbol{x}\in\mathcal{X}$ and parameter vector $\boldsymbol{\lambda}=\{\lambda_1,\ldots,\lambda_M\}$, CRFs define the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ of a particular output $\boldsymbol{y}\in\mathcal{Y}$ as being proportional to a product of potential functions on the cliques of a graph, which represents the interdependency of $\boldsymbol{y}$ and $\boldsymbol{x}$. That is:

$$p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda})=\frac{1}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x})}\prod_{c\in C(\boldsymbol{y},\boldsymbol{x})}\Phi_c(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\lambda})$$

where $\Phi_c(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\lambda})$ is a non-negative real value potential function on a clique $c\in C(\boldsymbol{y},\boldsymbol{x})$. $Z_{\boldsymbol{\lambda}}(\boldsymbol{x})=\sum_{\tilde{\boldsymbol{y}}\in\mathcal{Y}}\prod_{c\in C(\tilde{\boldsymbol{y}},\boldsymbol{x})}\Phi_c(\tilde{\boldsymbol{y}},\boldsymbol{x};\boldsymbol{\lambda})$ is a normalization factor over all output values, $\mathcal{Y}$.

Following the definitions of (Sha and Pereira, 2003), a log-linear combination of weighted features, $\Phi_c(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\lambda})=\exp(\boldsymbol{\lambda}\cdot\boldsymbol{f}_c(\boldsymbol{y},\boldsymbol{x}))$, is used as individual potential functions, where $\boldsymbol{f}_c$ represents a feature vector obtained from the corresponding clique c. That is, $\prod_{c\in C(\boldsymbol{y},\boldsymbol{x})}\Phi_c(\boldsymbol{y},\boldsymbol{x})=\exp(\boldsymbol{\lambda}\cdot F(\boldsymbol{y},\boldsymbol{x}))$, where $F(\boldsymbol{y},\boldsymbol{x})=\sum_c\boldsymbol{f}_c(\boldsymbol{y},\boldsymbol{x})$ is the CRF's global feature vector for $\boldsymbol{x}$ and $\boldsymbol{y}$.

The most probable output $\hat{\boldsymbol{y}}$ is given by $\hat{\boldsymbol{y}}=\arg\max_{\boldsymbol{y}\in\mathcal{Y}}p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda})$. However $Z_{\boldsymbol{\lambda}}(\boldsymbol{x})$ never affects the decision of $\hat{\boldsymbol{y}}$ since $Z_{\boldsymbol{\lambda}}(\boldsymbol{x})$ does not depend on $\boldsymbol{y}$. Thus, we can obtain the following discriminant function for CRFs:

$$\hat{\boldsymbol{y}}=\arg\max_{\boldsymbol{y}\in\mathcal{Y}}\boldsymbol{\lambda}\cdot F(\boldsymbol{y},\boldsymbol{x}). \tag{1}$$

The maximum (log-)likelihood (ML) of the conditional probability $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda})$ of training data $\{(\boldsymbol{x}^k,\boldsymbol{y}^{*k})\}_{k=1}^N$ w.r.t. parameters $\boldsymbol{\lambda}$ is the most basic CRF training criterion, that is, $\arg\max_{\boldsymbol{\lambda}}\sum_k\log p(\boldsymbol{y}^{*k}|\boldsymbol{x}^k;\boldsymbol{\lambda})$, where $\boldsymbol{y}^{*k}$ is the correct output for the given $\boldsymbol{x}^k$. Maximizing the conditional log-likelihood given by CRFs is equivalent to minimizing the log-loss function, $\sum_k-\log p(\boldsymbol{y}^{*k}|\boldsymbol{x}^k;\boldsymbol{\lambda})$. We minimize the following loss function for the ML criterion training of CRFs:

$$\mathcal{L}_{\boldsymbol{\lambda}}^{\mathrm{ML}}=\sum_k\Big[-\boldsymbol{\lambda}\cdot F(\boldsymbol{y}^{*k},\boldsymbol{x}^k)+\log Z_{\boldsymbol{\lambda}}(\boldsymbol{x}^k)\Big].$$

To reduce over-fitting, the Maximum a Posteriori (MAP) criterion of parameters $\boldsymbol{\lambda}$, that is, $\arg\max_{\boldsymbol{\lambda}}\sum_k\log p(\boldsymbol{\lambda}|\boldsymbol{y}^{*k},\boldsymbol{x}^k)\ \propto\ \sum_k\log p(\boldsymbol{y}^{*k}|\boldsymbol{x}^k;\boldsymbol{\lambda})p(\boldsymbol{\lambda})$, is now the most widely used CRF training criterion. Therefore, we minimize the following loss function for the MAP criterion training of CRFs:

$$\mathcal{L}_{\boldsymbol{\lambda}}^{\mathrm{MAP}}=\mathcal{L}_{\boldsymbol{\lambda}}^{\mathrm{ML}}-\log p(\boldsymbol{\lambda}). \tag{2}$$

There are several possible choices when selecting a prior distribution $p(\boldsymbol{\lambda})$. This paper only considers L_ϕ-norm prior, $p(\boldsymbol{\lambda})\propto\exp(-||\boldsymbol{\lambda}||^\phi/\phi C)$, which becomes a Gaussian prior when ϕ=2. The essential difference between ML and MAP is simply that MAP has this prior term in the objective function. This paper sometimes refers to the ML and MAP criterion training of CRFs as ML/MAP.

In order to estimate the parameters $\boldsymbol{\lambda}$, we seek a zero of the gradient over the parameters $\boldsymbol{\lambda}$:

$$\begin{aligned}\nabla\mathcal{L}_{\boldsymbol{\lambda}}^{\mathrm{MAP}}=-\nabla\log p(\boldsymbol{\lambda})+\sum_k\Bigg[&-F(\boldsymbol{y}^{*k},\boldsymbol{x}^k)\\&+\sum_{\boldsymbol{y}\in\mathcal{Y}^k}\frac{\exp(\boldsymbol{\lambda}\cdot F(\boldsymbol{y},\boldsymbol{x}^k))}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x}^k)}\cdot F(\boldsymbol{y},\boldsymbol{x}^k)\Bigg].\end{aligned} \tag{3}$$

The gradient of ML is Eq. 3 without the gradient term of the prior, $-\nabla\log p(\boldsymbol{\lambda})$.

The details of actual optimization procedures for linear chain CRFs, which are typical CRF applications, have already been reported (Sha and Pereira, 2003).

3 MCE Criterion Training for CRFs

The Minimum Classification Error (MCE) framework first arose out of a broader family of approaches to pattern classifier design known as *Generalized Probabilistic Descent (GPD)* (Katagiri et al., 1991). The MCE criterion minimizes an empirical loss corresponding to a smooth approximation of the classification error. This MCE loss is itself defined in terms of a *misclassification measure* derived from the *discriminant functions* of a given task. Via the smoothing parameters, the MCE loss function can be made arbitrarily close to the binary classification error. An important property of this framework is that it makes it

possible in principle to achieve the optimal Bayes error even under *incorrect* modeling assumptions. It is easy to extend the MCE framework to use evaluation measures other than the classification error, namely the linear combination of error rates. Thus, it is possible to optimize directly a variety of (smoothed) evaluation measures. This is the approach proposed in this article.

We first introduce a framework for MCE criterion training, focusing only on error rate optimization. Sec. 4 then describes an example of minimizing a different multivariate evaluation measure using MCE criterion training.

3.1 Brief Overview of MCE

Let $\boldsymbol{x} \in \mathcal{X}$ be an input, and $\boldsymbol{y} \in \mathcal{Y}$ be an output. The *Bayes decision rule* decides the most probable output $\hat{\boldsymbol{y}}$ for $\boldsymbol{x}$, by using the maximum a posteriori probability, $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}\in\mathcal{Y}} p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda})$. In general, $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda})$ can be replaced by a more general *discriminant function*, that is,

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}\in\mathcal{Y}} g(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\lambda}). \tag{4}$$

Using the discriminant functions for the possible output of the task, the *misclassification measure* $d()$ is defined as follows:

$$d(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda}) = -g(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda}) + \max_{\boldsymbol{y}\in\mathcal{Y}\setminus\boldsymbol{y}^*} g(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\lambda}). \tag{5}$$

where $\boldsymbol{y}^*$ is the correct output for $\boldsymbol{x}$. Here it can be noted that, for a given $\boldsymbol{x}$, $d() \geq 0$ indicates misclassification. By using $d()$, the minimization of the error rate can be rewritten as the minimization of the sum of 0-1 (step) losses of the given training data. That is, $\arg\min_{\boldsymbol{\lambda}} \mathcal{L}_{\boldsymbol{\lambda}}$ where

$$\mathcal{L}_{\boldsymbol{\lambda}} = \sum_k \delta(d(\boldsymbol{y}^{*k},\boldsymbol{x}^k,\boldsymbol{\lambda})). \tag{6}$$

$\delta(r)$ is a step function returning 0 if $r<0$ and 1 otherwise. That is, δ is 0 if the value of the discriminant function of the correct output $g(\boldsymbol{y}^{*k},\boldsymbol{x}^k,\boldsymbol{\lambda})$ is greater than that of the maximum *incorrect* output $g(\boldsymbol{y}^k,\boldsymbol{x}^k,\boldsymbol{\lambda})$, and δ is 1 otherwise.

Eq. 5 is not an appropriate function for optimization since it is a discontinuous function w.r.t. the parameters $\boldsymbol{\lambda}$. One choice of continuous misclassification measure consists of substituting 'max' with 'soft-max', $\max_k r_k \approx \log\sum_k \exp(r_k)$. As a result

$$d(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda}) = -g^* + \log\left[\mathcal{A}\sum_{\boldsymbol{y}\in\mathcal{Y}\setminus\boldsymbol{y}^*}\exp(\psi g)\right]^{\frac{1}{\psi}}, \tag{7}$$

where $g^* = g(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda})$, $g = g(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\lambda})$, and $\mathcal{A} = \frac{1}{|\mathcal{Y}|-1}$. ψ is a positive constant that represents L_ψ-norm. When ψ approaches ∞, Eq. 7 converges to Eq. 5. Note that we can design any misclassification measure, including non-linear measures for $d()$. Some examples are shown in the Appendices.

Of even greater concern is the fact that the step function δ is discontinuous; minimization of Eq. 6 is therefore NP-complete. In the MCE formalism, $\delta()$ is replaced with an approximated 0-1 loss function, $l()$, which we refer to as a *smoothing function*. A typical choice for $l()$ is the *sigmoid function*, $l^{\text{sig}}()$, which is differentiable and provides a good approximation of the 0-1 loss when the hyper-parameter α is large (see Eq. 8). Another choice is the *(regularized) logistic function*, $l^{\log}()$, that gives the upper bound of the 0-1 loss. Logistic loss is used as a conventional CRF loss function and provides convexity while the sigmoid function does not. These two smoothing functions can be written as follows:

$$\begin{aligned} l^{\text{sig}} &= (1+\exp(-\alpha\cdot d(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda})-\beta))^{-1} \\ l^{\log} &= \alpha^{-1}\cdot\log(1+\exp(\alpha\cdot d(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda})+\beta)), \end{aligned} \tag{8}$$

where α and β are the hyper-parameters of the training.

We can introduce a regularization term to reduce over-fitting, which is derived using the same sense as in MAP, Eq. 2. Finally, the objective function of the MCE criterion with the regularization term can be rewritten in the following form:

$$\mathcal{L}^{\text{MCE}}_{\boldsymbol{\lambda}} = \mathcal{F}_{l,d,g,\boldsymbol{\lambda}}\left[\{(\boldsymbol{x}^k,\boldsymbol{y}^{*k})\}_{k=1}^N\right] + \frac{||\boldsymbol{\lambda}||^\phi}{\phi C}. \tag{9}$$

Then, the objective function of the MCE criterion that minimizes the error rate is Eq. 9 and

$$\mathcal{F}^{\text{MCE}}_{l,d,g,\boldsymbol{\lambda}} = \frac{1}{N}\sum_{k=1}^N l(d(\boldsymbol{y}^{*k},\boldsymbol{x}^k,\boldsymbol{\lambda})) \tag{10}$$

is substituted for $\mathcal{F}_{l,d,g,\boldsymbol{\lambda}}$. Since N is constant, we can eliminate the term $1/N$ in actual use.

3.2 Formalization

We simply substitute the discriminant function of the CRFs into that of the MCE criterion:

$$g(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\lambda}) = \log p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda}) \propto \boldsymbol{\lambda}\cdot F(\boldsymbol{y},\boldsymbol{x}) \tag{11}$$

Basically, CRF training with the MCE criterion optimizes Eq. 9 with Eq. 11 after the selection of an appropriate misclassification measure, $d()$, and

smoothing function, $l()$. Although there is no restriction on the choice of $d()$ and $l()$, in this work we select sigmoid or logistic functions for $l()$ and Eq. 7 for $d()$.

The gradient of the loss function Eq. 9 can be decomposed by the following chain rule:

$$\nabla \mathcal{L}_{\boldsymbol{\lambda}}^{\text{MCE}} = \frac{\partial \mathcal{F}()}{\partial l()} \cdot \frac{\partial l()}{\partial d()} \cdot \frac{\partial d()}{\partial \boldsymbol{\lambda}} + \frac{||\lambda||^{\phi-1}}{C}.$$

The derivatives of $l()$ w.r.t. $d()$ given in Eq. 8 are written as: $\partial l^{\text{sig}}/\partial d = \alpha \cdot l^{\text{sig}} \cdot (1 - l^{\text{sig}})$ and $\partial l^{\text{log}}/\partial d = l^{\text{sig}}$.

The derivative of $d()$ of Eq. 7 w.r.t. parameters $\boldsymbol{\lambda}$ is written in this form:

$$\begin{aligned}\frac{\partial d()}{\partial \boldsymbol{\lambda}} = &- \frac{Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \psi)}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \psi) - \exp(\psi g^*)} \cdot F(\boldsymbol{y}^*, \boldsymbol{x}) \\ &+ \sum_{\boldsymbol{y} \in \mathcal{Y}} \left[\frac{\exp(\psi g)}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \psi) - \exp(\psi g^*)} \cdot F(\boldsymbol{y}, \boldsymbol{x}) \right]\end{aligned} \quad (12)$$

where $g = \boldsymbol{\lambda} \cdot F(\boldsymbol{y}, \boldsymbol{x})$, $g^* = \boldsymbol{\lambda} \cdot F(\boldsymbol{y}^*, \boldsymbol{x})$, and $Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \psi) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp(\psi g)$.

Note that we can obtain exactly the same loss function as ML/MAP with appropriate choices of $\mathcal{F}()$, $l()$ and $d()$. The details are provided in the Appendices. Therefore, ML/MAP can be seen as one special case of the framework proposed here. In other words, our method provides a generalized framework of CRF training.

3.3 Optimization Procedure

With linear chain CRFs, we can calculate the objective function, Eq. 9 combined with Eq. 10, and the gradient, Eq. 12, by using the variant of the forward-backward and Viterbi algorithm described in (Sha and Pereira, 2003). Moreover, for the parameter optimization process, we can simply exploit gradient descent or quasi-Newton methods such as L-BFGS (Liu and Nocedal, 1989) as well as ML/MAP optimization.

If we select $\psi = \infty$ for Eq. 7, we only need to evaluate the correct and the maximum incorrect output. As we know, the maximum output can be efficiently calculated with the Viterbi algorithm, which is the same as calculating Eq. 1. Therefore, we can find the maximum incorrect output by using the A* algorithm (Hart et al., 1968), if the maximum output is the correct output, and by using the Viterbi algorithm otherwise. It may be feared that since the objective function is not differentiable everywhere for $\psi = \infty$, problems for optimization would occur. However, it has been shown (Le Roux and McDermott, 2005) that even simple gradient-based (first-order) optimization methods such as GPD and (approximated) second-order methods such as *QuickProp* (Fahlman, 1988) and BFGS-based methods have yielded good experimental optimization results.

4 Multivariate Evaluation Measures

Thus far, we have discussed the error rate version of MCE. Unlike ML/MAP, the framework of MCE criterion training allows the embedding of not only a linear combination of error rates, but also any evaluation measure, including non-linear measures.

Several non-linear objective functions, such as F-score for text classification (Gao et al., 2003), and BLEU-score and some other evaluation measures for statistical machine translation (Och, 2003), have been introduced with reference to the framework of MCE criterion training.

4.1 Sequential Segmentation Tasks (SSTs)

Hereafter, we focus solely on CRFs in sequences, namely the linear chain CRF. We assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same length: $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$. In a linear chain CRF, y_i depends only on y_{i-1}.

Sequential segmentation tasks (SSTs), such as text chunking (Chunking) and named entity recognition (NER), which constitute the shared tasks of the *Conference of Natural Language Learning (CoNLL)* 2000, 2002 and 2003, are typical CRF applications. These tasks require the extraction of pre-defined segments, referred to as *target segments*, from given texts. Fig. 1 shows typical examples of SSTs. These tasks are generally treated as *sequential labeling problems* incorporating the IOB tagging scheme (Ramshaw and Marcus, 1995). The IOB tagging scheme, where we only consider the *IOB2* scheme, is also shown in Fig. 1. B-X, I-X and O indicate that the word in question is the beginning of the tag 'X', inside the tag 'X', and outside any target segment, respectively. Therefore, a segment is defined as a sequence of a few outputs.

4.2 Segmentation F-score Loss for SSTs

The standard evaluation measure of SSTs is the *segmentation F-score* (Sang and Buchholz, 2000):

$$\text{F}_\gamma = \frac{(\gamma^2 + 1) \cdot TP}{\gamma^2 \cdot FN + FP + (\gamma^2 + 1) \cdot TP} \quad (13)$$

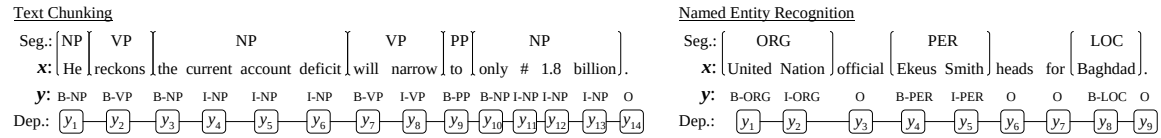

Figure 1: Examples of sequential segmentation tasks (SSTs): text chunking (Chunking) and named entity recognition (NER).

where TP, FP and FN represent true positive, false positive and false negative counts, respectively.

The individual evaluation units used to calculate TP, FN and PN, are not individual outputs y_i or output sequences $\boldsymbol{y}$, but rather segments. We need to define a *segment-wise loss*, in contrast to the standard CRF loss, which is sometimes referred to as an *(entire) sequential loss* (Kakade et al., 2002; Altun et al., 2003). First, we consider the point-wise decision w.r.t. Eq. 1, that is, $\hat{y}_i = \arg\max_{y_i \in \mathcal{Y}_1} g(\boldsymbol{y}, \boldsymbol{x}, i, \boldsymbol{\lambda})$. The point-wise discriminant function can be written as follows:

$$g(\boldsymbol{y}, \boldsymbol{x}, i, \boldsymbol{\lambda}) = \max_{\boldsymbol{y}' \in \mathcal{Y}_{|\boldsymbol{y}|}[y_i]} \boldsymbol{\lambda} \cdot F(\boldsymbol{y}', \boldsymbol{x}) \qquad (14)$$

where $\mathcal{Y}_j$ represents a set of all $\boldsymbol{y}$ whose length is j, and $\mathcal{Y}[y_i]$ represents a set of all $\boldsymbol{y}$ that contain y_i in the i'th position. Note that the same output $\hat{\boldsymbol{y}}$ can be obtained with Eqs. 1 and 14, that is, $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_n)$. This point-wise discriminant function is different from that described in (Kakade et al., 2002; Altun et al., 2003), which is calculated based on marginals.

Let $\boldsymbol{y}_{s_j}$ be an output sequence corresponding to the j-th segment of $\boldsymbol{y}$, where s_j represents a sequence of indices of $\boldsymbol{y}$, that is, $s_j = (s_{j,1}, \ldots, s_{j,|s_j|})$. An example of the Chunking data shown in Fig. 1, $\boldsymbol{y}_{s_4}$ is (B-VP, I-VP) where $s_4 = (7, 8)$. Let $\mathcal{Y}[\boldsymbol{y}_{s_j}]$ be a set of all outputs whose positions from $s_{j,1}$ to $s_{j,|s_j|}$ are $\boldsymbol{y}_{s_j} = (y_{s_{j,1}}, \ldots, y_{s_{j,|s_j|}})$. Then, we can define a segment-wise discriminant function w.r.t. Eq. 1. That is,

$$g(\boldsymbol{y}, \boldsymbol{x}, s_j, \boldsymbol{\lambda}) = \max_{\boldsymbol{y}' \in \mathcal{Y}_{|\boldsymbol{y}|}[\boldsymbol{y}_{s_j}]} \boldsymbol{\lambda} \cdot F(\boldsymbol{y}', \boldsymbol{x}). \qquad (15)$$

Note again that the same output $\hat{\boldsymbol{y}}$ can be obtained using Eqs. 1 and 15, as with the piece-wise discriminant function described above. This property is needed for evaluating segments since we do not know the correct segments of the test data; we can maintain consistency even if we use Eq. 1 for testing and Eq. 15 for training. Moreover, Eq. 15 obviously reduces to Eq. 14 if the length of all segments is 1. Then, the segment-wise misclassification measure $d(\boldsymbol{y}^*, \boldsymbol{x}, s_j, \boldsymbol{\lambda})$ can be obtained simply by replacing the discriminant function of the entire sequence $g(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\lambda})$ with that of segment-wise $g(\boldsymbol{y}, \boldsymbol{x}, s_j, \boldsymbol{\lambda})$ in Eq. 7.

Let $\boldsymbol{s}^{*k}$ be a segment sequence corresponding to the correct output $\boldsymbol{y}^{*k}$ for a given $\boldsymbol{x}^k$, and $\mathcal{S}(\boldsymbol{x}^k)$ be all possible segments for a given $\boldsymbol{x}^k$. Then, approximated evaluation functions of TP, FP and FN can be defined as follows:

$$TP_l = \sum_k \sum_{s_j^* \in \boldsymbol{s}^{*k}} \Big[1 - l(d(\boldsymbol{y}^{*k}, \boldsymbol{x}^k, s_j^*, \boldsymbol{\lambda}))\Big] \cdot \delta(s_j^*)$$

$$FP_l = \sum_k \sum_{s_j' \in \mathcal{S}(\boldsymbol{x}^k) \setminus \boldsymbol{s}^{*k}} l(d(\boldsymbol{y}^{*k}, \boldsymbol{x}^k, s_j', \boldsymbol{\lambda})) \cdot \delta(s_j')$$

$$FN_l = \sum_k \sum_{s_j^* \in \boldsymbol{s}^{*k}} l(d(\boldsymbol{y}^{*k}, \boldsymbol{x}^k, s_j^*, \boldsymbol{\lambda})) \cdot \delta(s_j^*)$$

where $\delta(s_j)$ returns 1 if segment s_j is a target segment, and returns 0 otherwise. For the NER data shown in Fig. 1, 'ORG', 'PER' and 'LOC' are the target segments, while segments that are labeled 'O' in $\mathbf{y}$ are not. Since TP_l should not have a value of less than zero, we select sigmoid loss as the smoothing function $l()$.

The second summation of TP_l and FN_l performs a summation over correct segments $\boldsymbol{s}^*$. In contrast, the second summation in FP_l takes all possible segments into account, but excludes the correct segments $\boldsymbol{s}^*$. Although an efficient way to evaluate all possible segments has been proposed in the context of semi-Markov CRFs (Sarawagi and Cohen, 2004), we introduce a simple alternative method. If we select $\psi = \infty$ for $d()$ in Eq. 7, we only need to evaluate the segments corresponding to the maximum incorrect output $\tilde{\boldsymbol{y}}$ to calculate FP_l. That is, $s_j' \in \mathcal{S}(\boldsymbol{x}^k) \setminus \boldsymbol{s}^{*k}$ can be reduced to $s_j' \in \tilde{\boldsymbol{s}}^k$, where $\tilde{\boldsymbol{s}}^k$ represents segments corresponding to the maximum incorrect output $\tilde{\boldsymbol{y}}$. In practice, this reduces the calculation cost and so we used this method for our experiments described in the next section.

Maximizing the segmentation F_γ-score, Eq. 13,

is equivalent to minimizing $\frac{\gamma^2 \cdot FN + FP}{(\gamma^2+1) \cdot TP}$, since Eq. 13 can also be written as $\mathrm{F}_\gamma = \frac{1}{1+\frac{\gamma^2 \cdot FN + FP}{(\gamma^2+1) \cdot TP}}$. Thus, an objective function closely reflecting the segmentation F_γ-score based on the MCE criterion can be written as Eq. 9 while replacing $\mathcal{F}_{l,d,g,\boldsymbol{\lambda}}$ with:

$$\mathcal{F}^{\text{MCE-F}}_{l,d,g,\boldsymbol{\lambda}} = \frac{\gamma^2 \cdot FN_l + FP_l}{(\gamma^2+1) \cdot TP_l}. \quad (16)$$

The derivative of Eq. 16 w.r.t. $l()$ is given by the following equation:

$$\frac{\partial \mathcal{F}^{\text{MCE-F}}_{l,d,g,\boldsymbol{\lambda}}}{\partial l()} = \begin{cases} \frac{\gamma^2}{Z_D} + \frac{(\gamma^2+1) \cdot Z_N}{Z_D^2}, & \text{if } \delta(s_j^*) = 1 \\ \frac{1}{Z_D}, & \text{otherwise} \end{cases}$$

where Z_N and Z_D represent the numerator and denominator of Eq. 16, respectively.

In the optimization process of the segmentation F-score objective function, we can efficiently calculate Eq. 15 by using the forward and backward Viterbi algorithm, which is almost the same as calculating Eq. 3 with a variant of the forward-backward algorithm (Sha and Pereira, 2003). The same numerical optimization methods described in Sec. 3.3 can be employed for this optimization.

5 Experiments

We used the same Chunking and 'English' NER task data used for the shared tasks of CoNLL-2000 (Sang and Buchholz, 2000) and CoNLL-2003 (Sang and De Meulder, 2003), respectively.

Chunking data was obtained from the Wall Street Journal (WSJ) corpus: sections 15-18 as training data (8,936 sentences and 211,727 tokens), and section 20 as test data (2,012 sentences and 47,377 tokens), with 11 different chunk-tags, such as NP and VP plus the 'O' tag, which represents the outside of any target chunk (segment).

The English NER data was taken from the Reuters Corpus2[1]. The data consists of 203,621, 51,362 and 46,435 tokens from 14,987, 3,466 and 3,684 sentences in training, development and test data, respectively, with four named entity tags, PERSON, LOCATION, ORGANIZATION and MISC, plus the 'O' tag.

5.1 Comparison Methods and Parameters

For ML and MAP, we performed exactly the same training procedure described in (Sha and Pereira, 2003) with L-BFGS optimization. For MCE, we only considered $d()$ with $\psi = \infty$ as described in Sec. 4.2, and used QuickProp optimization[2].

For MAP, MCE and MCE-F, we used the L_2-norm regularization. We selected a value of C from 1.0×10^n where n takes a value from -5 to 5 in intervals 1 by development data[3]. The tuning of smoothing function hyper-parameters is not considered in this paper; that is, α=1 and β=0 were used for all the experiments.

We evaluated the performance by Eq. 13 with $\gamma = 1$, which is the evaluation measure used in CoNLL-2000 and 2003. Moreover, we evaluated the performance by using the average sentence accuracy, since the conventional ML/MAP objective function reflects this sequential accuracy.

5.2 Features

As regards the basic feature set for Chunking, we followed (Kudo and Matsumoto, 2001), which is the same feature set that provided the best result in CoNLL-2000. We expanded the basic features by using bigram combinations of the same types of features, such as words and part-of-speech tags, within window size 5.

In contrast to the above, we used the original feature set for NER. We used features derived only from the data provided by CoNLL-2003 with the addition of character-level regular expressions of uppercases [A-Z], lowercases [a-z], digits [0-9] or others, and prefixes and suffixes of one to four letters. We also expanded the above basic features by using bigram combinations within window size 5. Note that we never used features derived from external information such as the Web, or a dictionary, which have been used in many previous studies but which are difficult to employ for validating the experiments.

5.3 Results and Discussion

Our experiments were designed to investigate the impact of eliminating the inconsistency between objective functions and evaluation measures, that is, to compare ML/MAP and MCE-F.

Table 1 shows the results of Chunking and NER. The $\mathrm{F}_{\gamma=1}$ and 'Sent' columns show the performance evaluated using segmentation F-score and

[1] http://trec.nist.gov/data/reuters/reuters.html

[2] In order to realize faster convergence, we applied online GPD optimization for the first ten iterations.

[3] Chunking has no common development set. We first train the systems with all but the last 2000 sentences in the training data as a development set to obtain C, and then retrain them with all the training data.

Table 1: Performance of text chunking and named entity recognition data (CoNLL-2000 and 2003)

		Chunking			NER		
	$l()$	n	$F_{\gamma=1}$	Sent	n	$F_{\gamma=1}$	Sent
MCE-F	(sig)	5	**93.96**	**60.44**	4	**84.72**	**78.72**
MCE	(log)	3	93.92	60.19	3	84.30	78.02
MCE	(sig)	3	93.85	60.14	3	83.82	77.52
MAP		0	93.71	59.15	0	83.79	77.39
ML		-	93.19	56.26	-	82.39	75.71

sentence accuracy, respectively. MCE-F refers to the results obtained from optimizing Eq. 9 based on Eq. 16. In addition, we evaluated the error rate version of MCE. MCE(log) and MCE(sig) indicate that logistic and sigmoid functions are selected for $l()$, respectively, when optimizing Eq. 9 based on Eq. 10. Moreover, MCE(log) and MCE(sig) used $d()$ based on $\psi{=}\infty$, and were optimized using QuickProp; these are the same conditions as used for MCE-F. We found that MCE-F exhibited the best results for both Chunking and NER. There is a significant difference ($p < 0.01$) between MCE-F and ML/MAP with the McNemar test, in terms of the correctness of both individual outputs, y_i^k, and sentences, $\boldsymbol{y}^k$.

NER data has 83.3% (170524/204567) and 82.6% (38554/46666) of 'O' tags in the training and test data, respectively while the corresponding values of the Chunking data are only 13.1% (27902/211727) and 13.0% (6180/47377). In general, such an imbalanced data set is unsuitable for accuracy-based evaluation. This may be one reason why MCE-F improved the NER results much more than the Chunking results.

The only difference between MCE(sig) and MCE-F is the objective function. The corresponding results reveal the effectiveness of using an objective function that is consistent as the evaluation measure for the target task. These results show that minimizing the error rate is not optimal for improving the segmentation F-score evaluation measure. Eliminating the inconsistency between the task evaluation measure and the objective function during the training can improve the overall performance.

5.3.1 Influence of Initial Parameters

While ML/MAP and MCE(log) is convex w.r.t. the parameters, neither the objective function of MCE-F, nor that of MCE(sig), is convex. Therefore, initial parameters can affect the optimization results, since QuickProp as well as L-BFGS can only find local optima.

Table 2: Performance when initial parameters are derived from MAP

		Chunking			NER		
	$l()$	n	$F_{\gamma=1}$	Sent	n	$F_{\gamma=1}$	Sent
MCE-F	(sig)	5	**94.03**	**60.74**	4	**85.29**	**79.26**
MCE	(sig)	3	93.97	60.59	3	84.57	77.71

The previous experiments were only performed with all parameters initialized at zero. In this experiment, the parameters obtained by the MAP-trained model were used as the initial values of MCE-F and MCE(sig). This evaluation setting appears to be similar to *reranking*, although we used exactly the same model and feature set.

Table 2 shows the results of Chunking and NER obtained with this parameter initialization setting. When we compare Tables 1 and 2, we find that the initialization with the MAP parameter values further improves performance.

6 Related Work

Various loss functions have been proposed for designing CRFs (Kakade et al., 2002; Altun et al., 2003). This work also takes the design of the loss functions for CRFs into consideration. However, we proposed a general framework for designing these loss function that included non-linear loss functions, which has not been considered in previous work.

With Chunking, (Kudo and Matsumoto, 2001) reported the best F-score of 93.91 with the voting of several models trained by *Support Vector Machine* in the same experimental settings and with the same feature set. MCE-F with the MAP parameter initialization achieved an F-score of 94.03, which surpasses the above result without manual parameter tuning.

With NER, we cannot make a direct comparison with previous work in the same experimental settings because of the different feature set, as described in Sec. 5.2. However, MCE-F showed the better performance of 85.29 compared with (McCallum and Li, 2003) of 84.04, which used the MAP training of CRFs with a feature selection architecture, yielding similar results to the MAP results described here.

7 Conclusions

We proposed a framework for training CRFs based on optimization criteria directly related to target multivariate evaluation measures. We first provided a general framework of CRF training based on MCE criterion. Then, specifically focusing on SSTs, we introduced an approximate segmentation F-score objective function. Experimental results showed that eliminating the inconsistency between the task evaluation measure and the objective function used during training improves the overall performance in the target task without any change in feature set or model.

Appendices

Misclassification measures

Another type of misclassification measure using soft-max is (Katagiri et al., 1991):

$$d(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\lambda}) = -g^* + \left[\mathcal{A} \sum_{y \in \mathcal{Y} \setminus y^*} g^{\psi} \right]^{\frac{1}{\psi}}.$$

Another $d()$, for g in the range $[0, \infty)$:

$$d(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\lambda}) = \left[\mathcal{A} \sum_{y \in \mathcal{Y} \setminus y^*} g^{\psi} \right]^{\frac{1}{\psi}} / g^*.$$

Comparison of ML/MAP and MCE

If we select $l^{\log}()$ with $\alpha = 1$ and $\beta = 0$, and use Eq. 7 with $\psi = 1$ and without the term $\mathcal{A}$ for $d()$. We can obtain the same loss function as ML/MAP:

$$\begin{aligned} & \log\left(1 + \exp(-g^* + \log(Z_{\boldsymbol{\lambda}} - \exp(g^*)))\right) \\ &= \log\left(\frac{\exp(g^*) + (Z_{\boldsymbol{\lambda}} - \exp(g^*))}{\exp(g^*)}\right) \\ &= -g^* + \log(Z_{\boldsymbol{\lambda}}). \end{aligned}$$

References

Y. Altun, M. Johnson, and T. Hofmann. 2003. Investigating Loss Functions and Optimization Methods for Discriminative Learning of Label Sequences. In *Proc. of EMNLP-2003*, pages 145–152.

S. E. Fahlman. 1988. An Empirical Study of Learning Speech in Backpropagation Networks. In *Technical Report CMU-CS-88-162, Carnegie Mellon University*.

S. Gao, W. Wu, C.-H. Lee, and T.-S. Chua. 2003. A Maximal Figure-of-Merit Approach to Text Categorization. In *Proc. of SIGIR'03*, pages 174–181.

P. E. Hart, N. J. Nilsson, and B. Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. on Systems Science and Cybernetics*, SSC-4(2):100–107.

M. Jansche. 2005. Maximum Expected F-Measure Training of Logistic Regression Models. In *Proc. of HLT/EMNLP-2005*, pages 692–699.

T. Joachims. 2005. A Support Vector Method for Multivariate Performance Measures. In *Proc. of ICML-2005*, pages 377–384.

B. H. Juang and S. Katagiri. 1992. Discriminative Learning for Minimum Error Classifi cation. *IEEE Trans. on Signal Processing*, 40(12):3043–3053.

S. Kakade, Y. W. Teh, and S. Roweis. 2002. An Alternative Objective Function for Markovian Fields. In *Proc. of ICML-2002*, pages 275–282.

S. Katagiri, C. H. Lee, and B.-H. Juang. 1991. New Discriminative Training Algorithms based on the Generalized Descent Method. In *Proc. of IEEE Workshop on Neural Networks for Signal Processing*, pages 299–308.

T. Kudo and Y. Matsumoto. 2001. Chunking with Support Vector Machines. In *Proc. of NAACL-2001*, pages 192–199.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of ICML-2001*, pages 282–289.

D. C. Liu and J. Nocedal. 1989. On the Limited Memory BFGS Method for Large-scale Optimization. *Mathematic Programming*, (45):503–528.

A. McCallum and W. Li. 2003. Early Results for Named Entity Recognition with Conditional Random Fields Feature Induction and Web-Enhanced Lexicons. In *Proc. of CoNLL-2003*, pages 188–191.

F. J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of ACL-2003*, pages 160–167.

Y. Qi, M. Szummer, and T. P. Minka. 2005. Bayesian Conditional Random Fields. In *Proc. of AI & Statistics 2005*.

L. A. Ramshaw and M. P. Marcus. 1995. Text Chunking using Transformation-based Learning. In *Proc. of VLC-1995*, pages 88–94.

J. Le Roux and E. McDermott. 2005. Optimization Methods for Discriminative Training. In *Proc. of Eurospeech 2005*, pages 3341–3344.

E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proc. of CoNLL/LLL-2000*, pages 127–132.

E. F. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proc. of CoNLL-2003*, pages 142–147.

S. Sarawagi and W. W. Cohen. 2004. Semi-Markov Conditional Random Fields for Information Extraction. In *Proc of NIPS-2004*.

F. Sha and F. Pereira. 2003. Shallow Parsing with Conditional Random Fields. In *Proc. of HLT/NAACL-2003*, pages 213–220.

B. Taskar, C. Guestrin, and D. Koller. 2004. Max-Margin Markov Networks. In *Proc. of NIPS-2004*.

I. Tsochantaridis, T. Joachims and T. Hofmann, and Y. Altun. 2005. Large Margin Methods for Structured and Interdependent Output Variables. *JMLR*, 6:1453–1484.