Integer Linear Programming Inference for Conditional Random Fields

Dan Roth DANR@UIUC.EDU
Wen-tau Yih YIH@UIUC.EDU
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA

Abstract

Inference in Conditional Random Fields and Hidden Markov Models is done using the Viterbi algorithm, an efficient dynamic programming algorithm. In many cases, general (non-local and non-sequential) constraints may exist over the output sequence, but cannot be incorporated and exploited in a natural way by this inference procedure. This paper proposes a novel inference procedure based on integer linear programming (ILP) and extends CRF models to naturally and efficiently support general constraint structures. For sequential constraints, this procedure reduces to simple linear programming as the inference process. Experimental evidence is supplied in the context of an important NLP problem, semantic role labeling.

1. Introduction

A large number of real world inference problems involve predicting values to sets of variables where complex and expressive structure can influence, or even dictate, what assignments are possible – predictions must respect constraints that could arise from the nature of the data or task specific conditions. For example, these problems are common in natural language processing tasks such as labeling words of a sentence with part-of-speech tags, identifying phrases in sentences or identifying and classifying the arguments of a verb in a sentence. In the latter task, for example, predictions must respect constraints such that "a verb cannot take two *subject* arguments" or "this verb does not take an *indirect object*," and others.

One of the most common approaches studied in the last few years to the problem of learning classifiers over structured output suggests to incorporate the dependencies among the variables into the learning process, and directly induces estimators that optimize a global performance measure. These solutions can be based on probabilistic models, as in Conditional Random Fields (CRFs) (Lafferty et al., 2001) – perhaps the most commonly used technique in this paradigm – or be discriminative as in (Collins, 2002; Taskar et al., 2004; Punyakanok et al., 2005).

In all these approaches, incorporating the dependencies is done by making Markovian assumptions among the output variables, which can be exploited via efficient inference algorithms. Specifically, CRFs can handle linear state sequences and tree structures elegantly and enjoys advantages of both generative and discriminative models. A rich set of feature functions can be used to model state sequences, capturing information on states, observations and state transitions. When used for predicting the state sequence, a dynamic programming algorithm, Viterbi, can be used to efficiently output the labels that maximize the joint conditional probability given the observation.

The efficiency of the CRF approach heavily depends on its first order Markov property – given the observation, the label of a token is assumed to depend only on the labels of its adjacent tokens. While this property does not need to hold for CRFs to be used in applications, it does prevent explicit modeling and exploiting more general constraints such as long distance dependencies. Indeed, recently Sarawagi and Cohen attempted to relax the Markov property to better tackle a phrase labeling problem (2005). This is a problem not only for training CRFs but also for incorporating additional constraints during inference. Although changing the transition matrices helps the Viterbi algorithm to handle certain types of constraints in evaluation (Kristjannson et al., 2004), so far there has not been any suggestion on how to handle more general constraints.

In this paper, we propose a novel inference procedure based on integer linear programming (ILP) to replace the Viterbi algorithm in the context of learning with structured output. The new model, ILP-CRF incor-

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

porates the ILP general optimization procedure seamlessly into the CRF model. This modeling allows one to add general constraints over the output space in a natural and systematic fashion. The constraint langauge is very expressive – general Boolean functions over the variables of interests can be expressed as linear (in)equalities. Yet, this modeling still allows, as we show, practically efficient solutions to large scale real world problems[1]. One of the nice properties of the proposed approach is that when no additional constraints are added, the problem reduces back to one that can be solved efficiently by linear programming. We experiment with this approach on an important large scale natural langauge problem, Semantic Role Labeling (Carreras & Màrquez, 2004) and exhibit significant improvement of the ILP-CRF over the standard CRF.

The rest of the paper is organized as follows. Sec. 2 briefly reviews CRFs models along with the Viterbi algorithm. The limitation of the Viterbi algorithm in incorporating constraints is discussed in Sec. 3. We present the ILP-CRF approach and discuss modeling problems in this paradigm in Sec. 4. Experimental results are presented in Sec. 5, along with a discussion of some general issues that pertain to efficiency and modularity on learning and maintaining structured output. Finally, Sec. 6 concludes this paper.

2. Conditional Random Fields

When used for sequential labeling problems, Conditional Random Fields are linear-chain Markov Random Fields that model $Pr(\mathbf{y}|\mathbf{x})$, where each node represents an element in the structured output $\mathbf{y}$ and the potential functions are decided by the observation through features over the input sequence $\mathbf{x}$.

Assume there are K feature functions, $f^1, \cdots, f^K$. Each of them maps a pair of sequence $(\mathbf{y}, \mathbf{x})$ and a token index i to $f^k(\mathbf{y}, \mathbf{x}, i) \in R$. The global feature vector is defined by

$$F(\mathbf{y}, \mathbf{x}) = \sum_i \langle f^1(\mathbf{y}, \mathbf{x}, i), \cdots, f^K(\mathbf{y}, \mathbf{x}, i) \rangle$$

Following (Lafferty et al., 2001; Sha & Pereira, 2003), the probability distribution is defined as

$$Pr_{\boldsymbol{\lambda}}(\mathbf{y}|\mathbf{x}) = \frac{\exp(\boldsymbol{\lambda} \cdot F(\mathbf{y}, \mathbf{x}))}{Z_{\boldsymbol{\lambda}}(\mathbf{x})}$$

where $\boldsymbol{\lambda}$ is the global weight vector, and $Z_{\boldsymbol{\lambda}}(\mathbf{x}) = \sum \exp(\boldsymbol{\lambda} \cdot F(\mathbf{y}, \mathbf{x}))$ is a normalization factor.

[1]Integer linear programming is NP-Hard; however, today's commercial packages can solve sparse problems with thousands of variables and constraints in a second, making this a realistic approach in many real world problems.

2.1. Inference

Given a chain-structured CRFs model, the general inference task is to find the the label sequence that maximizes the joint conditional probability, which can be calculated efficiently through the Viterbi algorithm. Let $\mathcal{Y}$ be the set of possible labels, where $|\mathcal{Y}| = m$. A set of $m \times m$ matrices $\{M_i(\mathbf{x}) | i = 0, \ldots, n-1\}$ is defined over each pair of labels $y', y \in \mathcal{Y}$

$$M_i(y', y|\mathbf{x}) = \exp(\sum_j \lambda_j f_j(y', y, \mathbf{x}, i)).$$

By augmenting two special nodes y_{-1} and y_n before and after the sequence with labels start and end respectively, the sequence probability is

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x})} \prod_{i=0}^{n} M_i(y_{i-1}, y_i|\mathbf{x}).$$

$Z(\mathbf{x})$ can be computed from the M_i's but not needed in evaluation. Therefore, we only need to find the label sequence $\mathbf{y}$ that maximizes the product of the corresponding elements of these $n+1$ matrices. The Viterbi algorithm is the standard method that computes the most likely label sequence given the observation. It *grows* the optimal label sequence gradually by scanning the matrices from position 0 to n. At step i, it records all the optimal sequences ending at a label $y, \forall y \in \mathcal{Y}$ (denoted by $\mathbf{y}_i^*(y)$), and also the corresponding product $P_i(y)$. The recursive function of this dynamic programming algorithm is as follows.

1. $P_0(y) = M_0(\mathsf{start}, y|\mathbf{x})$ and $\mathbf{y}_0^*(y) = y$
2. For $1 \leq i \leq n$, $\mathbf{y}_i^*(y) = \mathbf{y}_{i-1}^*(\hat{y}).(y)$ and $P_i(y) = \max_{y' \in \mathcal{Y}} P_{i-1}(y') M(y', y|\mathbf{x})$, where $\hat{y} = \text{argmax}_{y' \in \mathcal{Y}} P_{i-1}(y') M(y', y|\mathbf{x})$ and "." is the concatenation operator.

The optimal sequence is therefore $\mathbf{y}_{n-1}^* = [\mathbf{y}_n^*]_{0..n-1}$, which is the best path to the end symbol but taking only position 0 to position $n-1$.

2.2. Training

Training of CRFs requires estimating the values of the weight vector, $\boldsymbol{\lambda}$, which is usually done by maximizing the log-likelihood of a given training set $T = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^N$:

$$\mathcal{L}_{\boldsymbol{\lambda}} = \sum_k \log(p_{\boldsymbol{\lambda}}(\mathbf{y}_k|\mathbf{x}_k)) = \sum_k [\boldsymbol{\lambda} \cdot F(\mathbf{y}_k, \mathbf{x}_k) - \log Z_{\boldsymbol{\lambda}}(\mathbf{x}_k)]$$

Popular training methods include generalized iterative scaling, conjugate-gradient and limited-memory quasi-Newton. Interested readers may refer to (Sha & Pereira, 2003) for detailed comparison. In addition to the common maximum log-likelihood training, Collins (2002) suggests discriminatively learning the global weight vector by reducing the number of error predictions directly using the (voted) perceptron (Freund & Schapire, 1999). It iterates on each sequence, and updates the weight vector as follows.

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + F(\mathbf{y}_k, \mathbf{x}_k) - F(\hat{\mathbf{y}}_k, \mathbf{x}_k)$$

where $\hat{\mathbf{y}}_k$ is derived by the Viterbi inference algorithm based on the weight vector $\boldsymbol{\lambda}_t$. The algorithm cycles through the training data several times. The voted version reduces overfitting by using the average of the weight vectors during the training process as the final model parameters.

3. Incorporating Constraints in Viterbi

The Viterbi algorithm used for inference in CRFs can be extended, as is done with HMMs, to satisfy some types of additional constraints on the label sequence (e.g., constrained Viterbi described in (Kristjannson et al., 2004)). Consider, for example, NLP problems such as chunking (Tjong Kim Sang & Buchholz, 2000), semantic role labeling (Carreras & Màrquez, 2004), or information extraction (Kristjannson et al., 2004). In all these cases the task is to identify segments of consecutive words in the sentence and classify them to one of several classes. A word based representation called the BIO representation is often used for that purpose. The label B- (Begin) represents the first word of a segment, where - indicates the phrase type; I- (Inside) indicates that the word is part of, but not first in the segment, and the label O (Outside) is assigned to all other words in the sentence. When no two consecutive segments share the same type, the BIO representation can be simplified to the IO representation.

When an inference procedure like Viterbi is used, it is possible to modify the values of some elements in the matrices in order to enforce additional sequential constraints, beyond those encoded by the transition probabilities. For example, the BIO representation naturally disallows a label sequence that has an O label followed immediately by an I label. To avoid choosing this transition, the corresponding matrix entries can be set to 0 (or a very small number). Namely, $M_i(y_{i-1} = \mathsf{O}, y_i = \mathsf{I}|\mathbf{x}) = 0\ \forall i$ s.t $1 \le i \le n-1$, where n is the sentence's length.

Other types of enforced constraints may disallow some labels or ensure that some tokens are assigned some labels. For example, in an interactive information extraction scenario, the system may assume that the labels of some tokens are given by the user during evaluation (Kristjannson et al., 2004). This is easy to satisfy through a similar mechanism: if a token at the position i has to be labeled a, then no path is allowed to pass the state where $y_i \neq a$. That is, set $M_i(y_{i-1}, y_i)$ to be 0 for all $y_{i-1} \in \mathcal{Y}$ and all $y_i \in \mathcal{Y} - \{a\}$.

However, this matrix modification mechanism cannot be applied when the constraints define the relation of two distant tokens. One example of this type of constraints is the "no duplicate segments" in several tasks (e.g., semantic role labeling), where two different segments in a sentence cannot have the same label. Another example of a potential constraint in information extraction is "if FirstName appears in the sentence, then LastName must also appear." While these general constraints may not be exploited by the Viterbi algorithm, they can be easily represented as Boolean rules, and exploited using a different inference procedure based on integer linear programming.

4. Inference using ILP

The solution that Viterbi outputs is in fact the shortest path in the graph constructed as follows. Let n be the number of tokens in the sequence, and m be the number of labels each token can take. The graph consists of $nm+2$ nodes and $(n-1)m^2+2m$ edges. In addition to two special nodes start and end that denote the start and end positions of the path, the label of each token is represented by a node v_{ij}, where $0 \le i \le n-1$, and $0 \le j \le m-1$. If the path passes node v_{ij}, then label j is assigned to token i. For nodes that represent two adjacent tokens $v_{(i-1)j}$ and $v_{ij'}$, where $0 \le i \le n$, and $0 \le j, j' \le m-1$, there is a directed edge $x_{i,jj'}$ from $v_{(i-1)j}$ to $v_{ij'}$, with the cost $-\log(M_i(jj'|\mathbf{x}))$.

Obviously, the path from start to end will pass exactly one node on position i. That is, exactly one of the nodes $v_{i,j}, 0 \le j \le m-1$, will be picked. Figure 1 illustrates the graph. Suppose that $\mathbf{y} = y_0 y_1 \cdots y_{n-1}$ is the label sequence determined by the path. Then:

$$\operatorname*{argmin}_{\mathbf{y}} - \sum_{i=0}^{n-1} \log(M_i(y_{i-1}y_i|\mathbf{x})) = \operatorname*{argmax}_{\mathbf{y}} \prod_{i=0}^{n-1} M_i(y_{i-1}y_i|\mathbf{x}).$$

Namely, the nodes in the shortest path are exactly the labels returned by the Viterbi algorithm.

4.1. Solving General Shortest Path Problems Using ILP

A general shortest path problem can be reduced to an integer linear programming problem via the following straightforward setting (Wolsey, 1998). Given a directed graph $G = (V, E)$, two different nodes $s, t \in V$, and the nonnegative cost c_{uv} of each edge $(u, v) \in E$,

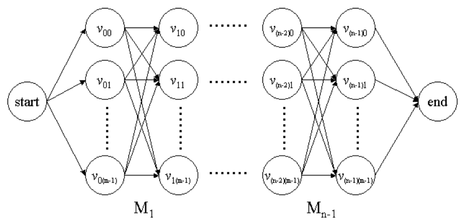

Figure 1. The graph that represents the labels of the tokens and the state transition (also known as the *trellis* in hidden Markov models)

we are seeking a path from s to t with the minimum cost. For each edge $(u, v) \in E$, we introduce an indicator variable x_{uv}. If (u, v) is in the minimum cost (shortest) path, then x_{uv} will be set to 1; otherwise, it will be 0. The cost function is therefore $\sum_{(u,v)\in E} c_{uv} \cdot x_{uv}$.

For each node in the graph except s and t, the number of inward edges in the path should be the same as the number of the outward ones. To nodes s and t, the differences of the numbers of inward edges and outward edges should be -1 and 1 respectively. If we use $V^-(v)$ to denote the nodes connected by the inward edges of v, and $V^+(v)$ to denote the nodes connected by the outward edges, then the complete (binary) integer linear program is:

$$\min \sum_{(u,v)\in E} c_{uv} \cdot x_{uv}$$

subject to:

$$\begin{aligned}
\sum_{u\in V^-(v)} x_{uv} - \sum_{w\in V^+(v)} x_{vw} &= 0, \qquad \forall v \in V - \{s,t\} \\
\sum_{u\in V^-(s)} x_{us} - \sum_{w\in V^+(s)} x_{sw} &= -1 \\
\sum_{u\in V^-(t)} x_{ut} - \sum_{w\in V^+(t)} x_{tw} &= 1 \\
x_{uv} \in \{0,1\}, &\qquad \forall (u,v) \in E
\end{aligned}$$

This integer linear programming representation for the shortest path problem can be solved directly by linear programming. In other words, even when the integer constraints (i.e., $x_{uv} \in \{0, 1\}, \forall(u, v) \in E$) are dropped, the optimal LP solution is still an integral solution. This is due to the theory of *unimodularity.*

Definition 1 (TU) A matrix $\mathbf{A}$ is *totally unimodular* if the determinant of every square submatrix of $\mathbf{A}$ is +1, -1, or 0.

Theorem 1 (Veinott & Dantzig) Let $\mathbf{A}$ be an (m, n)-integral matrix with full row rank m. Then the

$$\max \sum_{\substack{0 \le i \le n-1 \\ 0 \le y, y' \le m-1}} \log M_i(y, y') \cdot x_{i,yy'}$$

subject to:

$$\sum_{0\le y_1\le m-1} x_{i-1,y_1 y} - \sum_{0\le y_2\le m-1} x_{i,yy_2} = 0,$$

for all i, y such that $0 \le i \le n-1, 0 \le y \le m-1$.

$$\sum_{0\le y\le m-1} x_{-1,0y} = 1 \ \texttt{and} \sum_{0\le y\le m-1} x_{n,y0} = 1$$

$$x_{-1,0y}, x_{i,y_1 y}, x_{n,y0} \in \{0,1\},$$

for all i, y_1, y such that $0 \le i \le n-1, 0 \le y_1, y \le m-1$.

Figure 2. The integer linear program for solving the shortest problem described in Fig. 1. The positions for the start and end nodes are -1 and n. Variables $x_{-1,0y}$ and $x_{n,y0}$ represent the special edges connecting the start and end nodes respectively.

solution to the linear program $\max\{\mathbf{cx} : \mathbf{Ax} \le \mathbf{b}, \mathbf{x} \in R^n_+\}$ is integral for each integral vector $\mathbf{b}$, if and only if $\mathbf{A}$ is totally unimodular.

It can be shown that the coefficient matrix of the linear program for the shortest path problem is *totally unimodular* (Wolsey, 1998). Therefore, solving it takes only polynomial time using interior point algorithms for linear programming.

4.2. Replacing Viterbi with ILP

Following the formulation used in the linear-chain CRFs model, we can represent the corresponding shortest path problem using the ILP representation in Fig. 2. The power of this ILP formalism is its ability to represent expressive constraints in the output space. In fact, it is known that all possible Boolean functions over the variables of interest can be represented as sets of linear (in)equalities (Guéret et al., 2002); any constraint of interest can therefore be added to the basic program shown above. To illustrate this expressivity, we provide below a few detailed examples.

Example 1: In order to force the label of token i to be 0, we can add the following constraint:

$$\sum_{0\le y\le m-1} x_{i,y0} = 1.$$

Example 2: Consider the "no duplicate segment" constraint. We enforce it by making sure that a segment by type a that ends never starts again. Assuming label 0 means the O label in the IO representation, the constraint on the token level can be described by the following rule.

$$x_{i,ab}, a \neq b, 0 \Rightarrow \neg x_{j,ya}, i+1 \leq j \leq n-1.$$

The indicator variable $x_{i,ab}$ represents the antecedent, and $\overline{x_{i+1,ya}} \wedge \cdots \wedge \overline{x_{n-1,ya}}$ for all labels y (i.e., $0 \leq y \leq m-1$) are the consequence. Since $\overline{x} = 1 - x$ and the logic rule "$x \rightarrow x_1 \wedge x_2 \wedge \cdots \wedge x_n$" can be represented by the linear inequality "$nx \leq \sum_{i=i}^{n} x_i$", this constraint can be represented by the following set of linear inequalities:

$$m(n-1-i)x_{i,ab} \leq \sum_{\substack{0 \leq y \leq m-1 \\ i+1 \leq j \leq n-1}} 1 - x_{j,ya},$$

for all i, a, b such that $1 \leq i \leq n-2$, $1 \leq a \leq m-1$, $0 \leq b \leq m-1$, and $a \neq b$.

Example 3: The constraint: "if label a appears, then label b must also appear" can be represented using the following linear inequality.

$$\sum_{0 \leq y \leq m-1} x_{i,ya} \leq \sum_{\substack{0 \leq y \leq m-1 \\ 0 \leq j \leq n-1}} x_{j,yb}$$

for all i such that $0 \leq i \leq n-1$.

Example 4: When a segment $\mathcal{A}$ of tokens share the same label, then this constraint can be rephrased as "if a token $a \in \mathcal{A}$ is assigned label l, then all the tokens in $\mathcal{A}$ have to be l." Assume $\mathcal{A}$ ranges from tokens p to q. This constraint can be written as

$$v_{i,y} = \sum_{0 \leq y' \leq m-1} x_{i,y'y}, \quad (q-p)v_{p,l} \leq \sum_{p+1 \leq i \leq q} v_{i,l},$$

for all l s.t. $0 \leq l \leq m-1$. $v_{i,y}$ is a new binary variable that indicates whether token i is assigned label y.

Example 5: Finally, if we know that each sequence must have at least one segment of interest (i.e., not all tokens are the label O), this information can be encoded as follows:

$$\sum_{\substack{0 \leq i \leq n-1 \\ 0 \leq y \leq m-1}} x_{i,y0} \leq n-1$$

Note that all these constraints except the first one regulate the labels of distant tokens. This type of constraints cannot be satisfied by the Viterbi algorithm through the trick of modifying matrix elements, but can be easily represented in ILP. Although the new matrix may not be, in general, totally unimodular, and therefore linear programming relaxation does not guarantee an integer solution, commercial numerical packages usually solve this problem fairly efficiently given the size of the problem in practice. This is due the fact that despite the relatively large number of variables and constraints in our application, the constraints matrix is very sparse. In addition, if the optimal path in the original problem already satisfies the new constraint, adding this constraint will not make the problem harder.

We note that our work on this new inference procedure is inspired by (Punyakanok et al., 2004), which has shown that integer linear programming can be used efficiently in the context of large scale NLP problems. However, several differences in the formulation and modeling make our inference procedure a significant extension of previous work. In (Punyakanok et al., 2004), only hard constraints are incorporated to constrain the labels taken by local variables. The objective function is the expected number of correct local predictions, which is equivalent to the summation of local variables' conditional probabilities estimated by the classifiers. In addition, since there is no graph that describes the local constraints, the indicator variables introduced represent only the values of each segment variable rather than relations.

On the other hand, our inference procedure here provides an elegant way to *augment* CRFs with hard constraints, where the local and sequential constraint structure is still preserved. The objective function here maximizes the probability of the output sequence given the observation sequence. Moreover, indicator variables are used to describe the edges in the graph, and to facilitate the search of the shortest path, which corresponds to the most probable assignment.

5. Experiments

We conducted experiments on the *semantic role labeling* (SRL) task. This is an important and difficult natural language problem that attempts to discover the verb-argument structure for a given input sentence. The semantic argument identified for each verb in a given sentence may represent roles such as Agent, Patient or Instrument. We follow the definition of the PropBank (Kingsbury & Palmer, 2002) project and the CoNLL-2004 shared task definition (Carreras & Màrquez, 2004). Propbank defines six different types of arguments labelled as A0-A5 and AA, which have different semantics for each verb (as specified in the

I	**left**	**my**	**pearls**	**to**	**my**	**daughter-in-law**	**in**	**my**	**will**	**.**
I-A0	O	I-A1	I-A1	I-A2	I-A2	I-A2	O	O	O	O
A0	V	A1		A2						

Figure 3. The semantic role labels in the IO representation of an example sentence. The verb here is *left*.

PropBank Frame files) and a number of other types of modifiers and adjuncts. Figure 3 shows the SRL label sequence of the sentence, " I *left* my pearls to my daughter-in-law in my will." Here A0 represents the *leaver*, A1 represents the *thing left* and A2 represents the *benefactor*. Among the full set of PropBank labels, we only consider in the current experiments the core arguments, namely Arg0, Arg1, ..., Arg5.

For each verb in a sentence, a semantic argument is a segment of consecutive chunks (atomic phrases), and no two arguments share the same label. Therefore, we use the IO representation described in Sec. 3. The goal is to assign each chunk with one of the following labels: O, I-A0, I-A1, I-A2, I-A3, I-A4, and I-A5, as in Fig. 3.

Since the arguments of a given verb do not overlap, we can easily cast the task as a sequence labeling problem. To reduce the length of the sequence, we first transform the data from the original word-based format to a chunk-based format, as suggested by Hacioglu et al. (2004). In this representation, the basic token is a chunk, and its spelling and part-of-speech (POS) tag are replaced by those of its head word.

The state features (features of each chunk) we choose are a subset of features that are used in most SRL systems (Carreras & Màrquez, 2004), which include: *word*, *pos*, *chunk type*, *verb's pos*, *verb's lemma form*, *verb's voice* (active or passive), *position* (whether the target chunk is before or after the verb), *chunk path* (the chunk label sequence from the predicate to the target chunk), *clause path* (the path from the verb to the target chunk following clauses and chunks), *position relative to the verb* (whether the target chunk and verb are in the same clause), *verb class*, and *named entity*. In addition, features *word*, *pos*, *chunk type* of the neighboring chunks (with window size -2 to +2) are also extracted. Other standard and transition features used in the CRFs model include: *edge* (the current label and the previous label), *start* (whether the current label can be a start state or not), and *end* (whether the current label can be an end state or not).

The data we use is provided by the CoNLL-2004 shared task of semantic-role labeling (Carreras & Màrquez, 2004), which is a subset of the PropBank corpus. The training set covers sections 15-18 and the testing set is section 20.

5.1. Applying General Constraints

We first observe the effect of applying general constraints in the proposed inference procedure for CRFs. In this set of experiments, we train two basic CRF models using the maximum log-likelihood approach (Quasi-Newton optimization algorithm, LBFGS; with Gaussian prior $\sigma^2 = 0.01$) and the discriminative method (based on voted perceptron, suggested by Sha and Pereira (2003); Collins (2002))[2]. The number of training iterations is 100 for both methods.

General constraints are not used in training but only in evaluation. In particular, we would like to know to what extent the additional constraints, which cannot be modeled in the standard Viterbi inference, can improve the SRL predictions. We incrementally add the following constraints, modeled via the ILP based inference procedure.

1. **No duplicate argument labels**: In the SRL task, a verb in a sentence cannot have two arguments of the same type.
2. **Argument candidates**: Following the heuristic suggested by Gueret and Palmer (2004), we can generate a candidate list with high recall but low precision. Each candidate argument is a segment of consecutive chunks. Although not every candidate is an argument of the target verb, each chunk in the candidate has to be assigned the same label. This is an effective constraint that provides argument-level information.
3. **At least one argument**: Since we know that each verb in a sentence must have at least one core argument, at least one chunk will be assigned a label other than O.
4. **Known verb position**: Because the position of the active verb in the sentence is given, this knowledge can be added as one additional constraint – for chunks that are the active verbs, we know in advance that the labels are O.
5. **Disallow arguments**: Given a particular verb, not every argument type is legitimate. The arguments that a verb can take are defined in the *frame* files in the PropBank corpus.

[2]We build our systems based on the CRF package developed by Sarawagi (2004).

		CRF-ML			CRF-D		
		Rec	Prec	F_1	Rec	Prec	F_1
	basic	62.53	70.91	66.46	66.64	71.83	69.14
1	+ no dup	62.52	72.41	67.10	66.21	73.66	69.74
2	+ candidate	65.61	79.23	71.78	68.64	79.44	73.64
3	+ argument	66.54	77.76	71.71	69.42	78.57	73.71
4	+ verb pos	66.56	77.75	71.72	69.52	78.59	73.78
5	+ disallow	66.70	78.08	**71.94**	69.62	78.76	**73.91**

Table 1. The overall recall, precision, and F_1 of two SRL systems. Learning algorithms include: CRF-ML (maximum log-likelihood), CRF-D (discriminative).

		VP	VW	P	W
	basic	58.15	54.32	53.03	50.78
1	+ no dup	64.33	61.87	60.59	59.13
2	+ candidate	74.17	71.72	70.03	70.20
3	+ argument	74.02	71.76	69.98	70.32
4	+ verb pos	74.03	71.84	70.05	70.42
5	+ disallow	**74.49**	72.04	70.36	70.67

Table 2. The overall F_1 of SRL systems based on local learning. Learning algorithms include: VP (voted perceptron), VW (voted winnow), P (perceptron), W (winnow).

Among the above constraints, only (4) and (5) can be exploited by the constrained Viterbi algorithm. The other constraints regulate the labels of distant tokens without specifying which labels are not allowed or must be assigned. Table 1 shows the performance in recall, precision and F-measure. As is clearly shown in the table, these general constraints improve the results significantly, for both CRF training algorithms.

At this stage, our approach is able to acquire the best global assignment $y^* = \operatorname{argmax} p(\mathbf{y}|\mathbf{x})$ and its score using the current model parameters, but not the marginal probability $p(y|\mathbf{x})$, where y is the individual token and $\mathbf{x}$ is the observation. While this implies that our inference procedure cannot be incorporated in the maximum log-likelihood training (CRF-ML), it still allows us to interleave inference and training in the discriminative method (CRF-D). Although the training process is more time-consuming, the result is not satisfactory. By adding all 5 additional constraints in both training and testing, the F-measure is 69.82% – about 4% lower than the basic CRF-D plus inference with 5 constraints. One explanation is that the inference-based training method may require more training examples (Punyakanok et al., 2005).

5.2. Local Training and Inference

Given the significant impact of the post-learning inference, we decided to evaluate also the approach of decoupling learning and inference completely. That is, a local multi-class classifier is trained to predict the label of each token as one of O, I-A0, I-A1, I-A2, I-A3, I-A4, and I-A5. During learning, it has no knowledge of the sequential constraints nor the global hard constraints. Only at evaluation time the hard constraints were incorporated through inference. The learning algorithms we used are regularized versions of perceptron, winnow, voted perceptron and voted winnow[3].

[3]We use the version implemented in SNoW (Carlson et al., 1999), which can be downloaded from http://l2r.cs.uiuc.edu/ cogcomp/.

The regularization is done through the idea similar to margin perceptron. The weights are updated not only when it makes mistake, but also when the points lie within a certain distance (margin) from the hyperplane. The voted version of perceptron and winnow outputs the averaged weight vector during training instead of the final parameters.

Table 2 shows the performance of the local learning algorithms, with the effect of incorporating these constraints incrementally listed in each following row. One of the most interesting observations from the experimental results is the dramatic performance difference of the local learning approaches. The basic performance of the four systems based on local training (i.e., (voted) perceptron and (voted) winnow) is significantly worse than the two CRF systems. This is not surprising given that in CRFs, the information on state transition is encoded as features and has direct influence on the potential function. However, as more constraints are added to the inference procedure, this new information, available to these algorithms only at evaluation time, boosts the performance of the systems based on local learning, and shrinks the performance gap. In fact, when all five constraints are incorporated via the inference, locally trained voted perceptron generates the best performance among all systems. These results are even more significant if we consider the training efficiency of the different systems, shown in Table 3. It is well known that global training is a time consuming task, as is shown for the CRF entries in the table. The essential reason is that inference needs to be done multiple times. On the other hand, training local learning algorithms such as perceptron and winnow can be done very efficiently. Local training has other advantages from the perspective of modularity, and there is no need to know the constraints at training time. It is therefore important to realize, as we show, that in the presence of general constraints, global training and local training provide comparable results. We investigate this issue more thoroughly in a companion paper (Punyakanok et al., 2005).

	CRF-ML	CRF-D	CRF-D (IBT)	VP	VW
Time (hrs)	48	38	145	0.8	0.8

Table 3. The training time of different SRL systems in hours. CRF-D (IBT) is the inference based training that interleaves discriminative learning and our inference procedure. A Xeon 2.6GHz machine with 6GB memory is used for all the experiments. The number of features is 810,451.

6. Conclusions

Although the Viterbi inference algorithm used in CRFs provides a good way to incorporate sequential constraints in training structured output predictors, its ability to incorporate additional general constraints over the output space is limited. This paper has developed an enhanced inference procedure, in which general constraint structure can be encoded at evaluation time, and can be incorporated as part of the discriminative training algorithm. We have shown that the shortest path problem solved by the Viterbi algorithm can be represented and solved through integer linear programming. While for the standard shortest path problem the corresponding integer linear program can be solved efficiently by linear programming relaxation, this formalism provides a natural and systematic way to incorporate general constraints and solve them in the same way. This novel inference method was tested on an important sequence labeling problem – semantic role labeling, exhibiting significant improvement by adding constraints that the Viterbi algorithm cannot take. In addition, we have discussed efficiency issues; we observed that, sometimes, in the presence of structure on the output, enforcing the constraints only at the evaluation time results in comparable performance at a much lower cost.

Acknowledgments

We thank Vasin Punyakanok for valuable discussions and the anonymous reviewers for their helpful comments. We are also grateful to Dash Optimization for the free academic use of Xpress-MP. This research is supported by NFS grant ITR IIS-0085836, a DOI grant under the Reflex program, and an ONR MURI Award.

References

Carlson, A., Cumby, C., Rosen, J., & Roth, D. (1999). *The SNoW learning architecture* (Technical Report UIUCDCS-R-99-2101). Dept. of CS, UIUC.

Carreras, X., & Màrquez, L. (2004). Introduction to the CoNLL-2004 shared task: Semantic role labeling. *Proc. of CoNLL-2004*.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *Proc. of EMNLP-2002*.

Freund, Y., & Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, *37*, 277–296.

Guéret, C., Prins, C., & Sevaux, M. (2002). *Applications of optimization with Xpress-MP*. Dash Optimization. Translated and revised by Susanne Heipcke.

Gueret, N., & Palmer, M. (2004). Calibrating features for semantic role labeling. *Proc. of the EMNLP-2004*

Hacioglu, K., Pradhan, S., Ward, W., Martin, J. H., & Jurafsky, D. (2004). Semantic role labeling by tagging syntactic chunks. *Proc. of CoNLL-2004*.

Kingsbury, P., & Palmer, M. (2002). From Treebank to PropBank. *Proc. of LREC-2002*. Spain.

Kristjannson, T., Culotta, A., Viola, P., & McCallum, A. (2004). Interactive information extraction with constrained conditional random fields. *Proc. of AAAI-2004*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. of ICML-2001*.

Punyakanok, V., Roth, D., Yih, W., & Zimak, D. (2004). Semantic role labeling via integer linear programming inference. *Proc. of COLING-2004*.

Punyakanok, V., Roth, D., Yih, W., & Zimak, D. (2005). Learning and inference over constrained output. *Proc. of IJCAI-2005*.

Sarawagi, S. (2004). Conditional Random Field (CRF) Package. http://crf.sourceforge.net/.

Sarawagi, S., & Cohen, W. W. (2005). Semi-markov conditional random fields for information extraction. *Proc. of NIPS-2004*.

Sha, F., & Pereira, F. (2003). Shallow parsing with conditional random fields. *HLT-NAACL-2003*

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin markov networks. *Proc. of NIPS-2003*.

Tjong Kim Sang, E. F., & Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. *Proc. of CoNLL-2000*.

Wolsey, L. (1998). *Integer programming*. John Wiley & Sons, Inc.