Structured Output Learning with Indirect Supervision

Ming-Wei Chang MCHANG21@ILLINOIS.EDU
Vivek Srikumar VSRIKUM2@ILLINOIS.EDU
Dan Goldwasser GOLDWAS1@ILLINOIS.EDU
Dan Roth DANR@ILLINOIS.EDU
Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA

Abstract

We present a novel approach for structure prediction that addresses the difficulty of obtaining labeled structures for training. We observe that structured output problems often have a companion learning problem of determining whether a given input possesses a good structure. For example, the companion problem for the part-of-speech (POS) tagging task asks whether a given sequence of words has a corresponding sequence of POS tags that is "legitimate". While obtaining *direct supervision* for structures is difficult and expensive, it is often very easy to obtain *indirect supervision* from the companion binary decision problem.

In this paper, we develop a large margin framework that jointly learns from both direct and indirect forms of supervision. Our experiments exhibit the significant contribution of the easy-to-get indirect binary supervision on three important NLP structure learning problems.

1. Introduction

Making complex decisions in real world problems often involves predicting values to sets of interdependent variables. These problems, called *structured output* problems have attracted considerable interest in the machine learning community over the last few years (e.g., (Lafferty et al., 2001; Tsochantaridis et al., 2004; Taskar et al., 2004; Roth & Yih, 2007) and many others).

Many tasks in Natural Language Processing (NLP) and Computer Vision can be naturally modeled as structure learning problems. Consider, for example, the problem of *entity transliteration*. Given an English name (e.g., `Italy`), generate the corresponding name in Hebrew (pronounced `Ee'Tal'Ya`). To do so, there is a need to learn a model which, given a pair {English named entity, its Hebrew transliteration}, finds the phonetic (character sequence) alignment between them. Solving these problems often involves both learning and inference steps – for an input, there is a need to search a complex output space and generate the best *feasible* element in it. In the above example, most phonetic alignments are not legitimate. Unfortunately, the natural complexity of the output space makes it expensive to obtain labeled data and provide *direct supervision* for the target structure prediction task.

In this paper, we formalize the observation that many structured output prediction problems have a *companion* binary decision problem of predicting whether an input can produce a good structure or not. For instance, consider the following binary problem (Klementiev & Roth, 2008): given Named Entities (NE) in two languages, determine if they represent transliterations of each other. Since transliterations of NEs should sound similar, this binary decision problem can be formulated as the following question: "Can the two NEs produce good phonetic sequence alignment?" This binary decision task determines whether a given input (here, a pair of English and Hebrew NEs) can generate a well-formed structure (in this case, the mapping).

Our work is motivated by several recent works (Chang et al., 2009; Felzenszwalb et al., 2009; Chang et al., 2010) which solve *binary decision problems* by learning to predict a (latent) structure on which the binary labels depend. We observe that it is often easy to obtain supervision (binary labels) for the companion problem, and we refer to the supervision for the companion binary task as *indirect supervision* for the target structure prediction task. The key research question addressed in this paper is: *Can the structured output prediction task benefit from* ***indirect supervision*** *to the companion problem?*

We propose a novel framework for Joint Learning with Indirect Supervision (**J-LIS**) which uses both structured and binary labeled data (Sec. 2). With this framework, one can learn from a very small number of structured labeled examples, which are hard to come by, and gain substantially

Appearing in *Proceedings of the 27^{th} International Conference on Machine Learning*, Haifa, Israel, 2010.

from indirect binary supervision for the companion decision problem, that, as we show, is easy to obtain.

We develop a learning algorithm for this formulation that generalizes the structured output SVM by *jointly* learning from both forms of supervision (Sec. 3). Moreover, our optimization algorithm allows incorporation of constraints on the output space, an approach that is often found very useful in structured output problems. We experimentally demonstrate the effectiveness of our algorithm in three different structured output prediction domains—transliteration, information extraction and part-of-speech tagging (Sec. 4). In all domains, indirect supervision is easily obtainable and significantly improves performance on the target structured output prediction task.

2. Learning with Indirect Supervision

In many important applications, a companionship relation exists between a structured output prediction task and a binary decision task. We start our explanation of the learning framework with a few motivating examples.

Information Extraction An advertisement on Craiglist contains fields like *size, rent* and *location*. Extracting the fields from text is a sequence tagging problem. A companion task is to determine if an article is a well-formed advertisement. The relation between these tasks is clear; the binary task asks whether advertisement fields can be extracted from the article. The labeled data for the binary problem is easy to obtain, for example by crawling the web, and generating negative data by shuffling their contents[1].

Object Part Recognition Consider the computer vision task of labeling the "parts" (e.g., wheels) of a car in an image. A companion binary task can be defined as predicting whether an image contains a car. The relationship is clear, as only an image containing a car will also contain car parts in the right position. While labeling parts in an image is difficult, obtaining car and non-car images is easy.

Typically, structured output learning uses *direct supervision* consisting of annotated structures. We incorporate binary labeled examples for the companion task into the learning process as *indirect supervision*.

2.1. Model

We now formalize the intuition described above. First, we introduce the notation. Let $S = \{(\mathbf{x}_i, \mathbf{h}_i)\}_{i=1}^{l}$ denote the direct supervision set consisting of l examples $\mathbf{x}_i$ and their corresponding structures $\mathbf{h}_i$. Likewise, let $B = \{(\mathbf{x}_i, y_i)\}_{i=l+1}^{l+m}$ denote the *indirect supervision* set, where, each $y_i \in \{1, -1\}$. For brevity, we write $i \in S$ to indicate $(\mathbf{x}_i, \mathbf{h}_i) \in S$ and $i \in B$ to indicate $(\mathbf{x}_i, y_i) \in B$. We denote B^+ and B^- as a partition of B consisting of positive and negative instances of B respectively. For any $\mathbf{x}$, $\mathcal{H}(\mathbf{x})$ denotes the set of all feasible structures. Let $\Phi(\mathbf{x}, \mathbf{h})$ be a feature generation function. We define $\mathcal{X}$ as the set of all feature vectors for $\mathbf{x}$. That is, $\mathcal{X} = \{\Phi(\mathbf{x}, \mathbf{h}) \mid \mathbf{h} \in \mathcal{H}(\mathbf{x})\}$.

In the standard structured output prediction task, the goal of learning is to find a weight vector $\mathbf{w}$ that, for every example $(\mathbf{x}_i, \mathbf{h}_i) \in S$, assigns the highest score to the correct element $\mathbf{h}_i$ of $\mathcal{H}(\mathbf{x}_i)$. That is, we wish to find $\mathbf{w}$ such that,

$$\mathbf{h}_i = \arg\max_{\mathbf{h} \in \mathcal{H}(\mathbf{x})} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}).$$

This *inference problem* can be solved by specific algorithms or a general framework such as integer linear programming (Roth & Yih, 2005). To use indirect supervision, our key assumption is that an input $\mathbf{x}$ is valid (its $y = +1$) if and only if its best structure is well-formed. Conversely, the input is invalid (its $y = -1$) if every structure for that input is bad. We require that the weight vector $\mathbf{w}$ should satisfy two conditions:

$$\begin{aligned} &\forall\ (\mathbf{x}, -1) \in B^-, \quad \forall \mathbf{h} \in \mathcal{H}(\mathbf{x}), \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{h}) \leq 0, \\ &\forall\ (\mathbf{x}, +1) \in B^+, \quad \exists \mathbf{h} \in \mathcal{H}(\mathbf{x}), \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{h}) \geq 0 \end{aligned} \quad (1)$$

The geometric interpretation of this setting is shown in Fig. 1. Circles represent the set $\mathcal{X} = \{\Phi(\mathbf{x}, \mathbf{h}) \mid \mathbf{h} \in \mathcal{H}(\mathbf{x})\}$, the feature vectors corresponding to feasible structures of an example $\mathbf{x}$. The vector $\mathbf{w}$ defines a hyperplane separating the examples into positive and negative classes. The figure on the left denotes standard structured output learning, where $\mathbf{w}$ is learned using a labeled set S. A positive example, $\mathbf{x}$ has a well defined structure, but our prediction is incorrect. On the right, the learning algorithm uses indirect supervision. Following the requirement that structures obtained from negative examples should have negative scores, the weight vector $\mathbf{w}$ is pushed into a better region, allowing the structure predictor to predict the correct structure.

2.2. Learning

In the standard structural SVM, the goal of learning is to solve the following minimization problem:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C_1 \sum_{i \in S} L_S(\mathbf{x}_i, \mathbf{h}_i, \mathbf{w}) \quad (2)$$

where, $L_S(\mathbf{x}_i, \mathbf{h}_i, \mathbf{w})$ represents the loss function for the structure prediction. The function L_S can be written as

$$L_S(\mathbf{x}_i, \mathbf{h}_i, \mathbf{w}) = \ell \left(\max_{\mathbf{h}} \left(\Delta(\mathbf{h}, \mathbf{h}_i) - \mathbf{w}^T \Phi_{\mathbf{h}_i, \mathbf{h}}(\mathbf{x}_i) \right) \right) \quad (3)$$

where $\Phi_{\mathbf{h}_i, \mathbf{h}}(\mathbf{x}_i) = \Phi(\mathbf{x}_i, \mathbf{h}_i) - \Phi(\mathbf{x}_i, \mathbf{h})$ and Δ is a function which returns the distance between two structures. In this paper, we define Δ as the Hamming distance between structures, but our algorithm can be used with any suitable

[1] Shuffling should also be treated as a supervision source, since we *know* it will generate ill-formed examples for this domain.

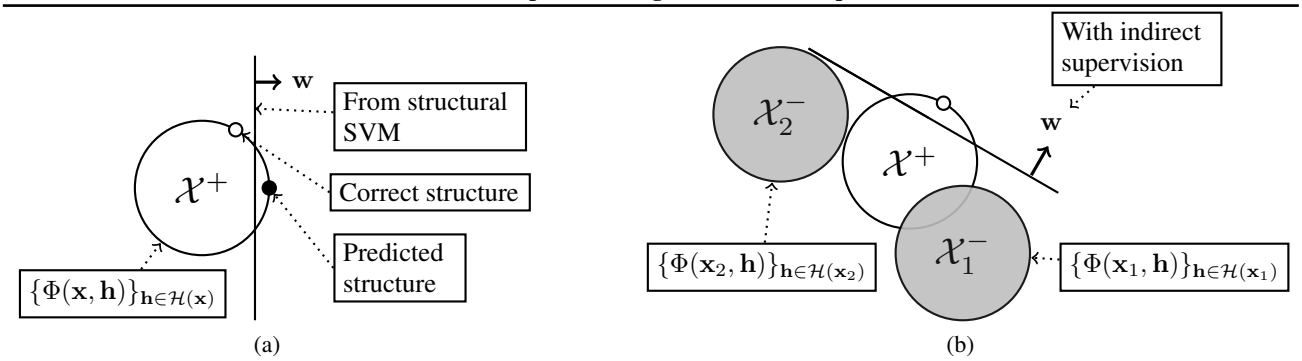

Figure 1. Learning with indirect supervision when the target output is H. Each circle represents the set of feature vectors of feasible structures of an example and $\mathbf{w}$ denotes a hyperplane. **(a)** Suppose we have learned a $\mathbf{w}$ using a structured labeled set S. For a positive example, $\mathbf{x} \in B^+$, we know there exists a well defined, unknown structure, but our prediction is incorrect. **(b)** After adding two negative examples: Negative examples, by definition, do not have a well formed structure. That is, *every* structure for $\mathbf{x}_1, \mathbf{x}_2 \in B^-$ should be scored below a threshold, and *some* structure of $\mathbf{x}$ should score above it. The negative examples restrict the space of hyperplanes supporting the right decision for $\mathbf{x}$. See Section 2 for details.

definition for the distance between structures. The function $\ell : \mathbb{R} \to \mathbb{R}$ can be instantiated by many commonly used loss functions such as hinge loss, with $\ell(x) = \max(0, x)$, and squared-hinge loss, with $\ell(x) = \max(0, x)^2$.

We incorporate indirect supervision using the intuition from Eq. (1) to define the problem of **Joint Learning with Indirect Supervision (J-LIS)**:

Given a structured labeled dataset S and a binary labeled dataset B, the goal of learning is to find $\mathbf{w}$ that minimizes the objective function $Q(\mathbf{w})$, which is defined as

$$\frac{\|\mathbf{w}\|^2}{2} + C_1 \sum_{i\in S} L_S(\mathbf{x}_i, \mathbf{h}_i, \mathbf{w}) + C_2 \sum_{i\in B} L_B(\mathbf{x}_i, y_i, \mathbf{w}), \quad (4)$$

where L_S, as before, is the loss for the structure prediction, and L_B is the loss for the binary prediction.

The intuition from Eq. (1) is incorporated into the binary loss L_B. The first inequality of Eq. (1) is equivalent to requiring the highest scoring structure of all negative examples to be below the threshold. The second inequality will be satisfied if the highest scoring structure of every positive example is above the threshold. Thus, the two inequalities can be re-stated as follows:

$$\forall\ (\mathbf{x}, -1) \in B^-, \quad \max_{\mathbf{h}\in\mathcal{H}(\mathbf{x})} \mathbf{w}^T \Phi(\mathbf{x}', \mathbf{h}) \leq 0,$$
$$\forall\ (\mathbf{x}, +1) \in B^+, \quad \max_{\mathbf{h}\in\mathcal{H}(\mathbf{x})} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{h}) \geq 0.$$

Furthermore, the design of the binary loss needs to account for the fact that different examples can be of different sizes. Hence, the weight vectors $\Phi(\mathbf{x}_i, \mathbf{h})$ need to be normalized according to the size of the input. Accordingly, using $\kappa(\mathbf{x})$ as normalization for an input $\mathbf{x}$, we define the L_B as[2]

$$L_B(\mathbf{x}_i, y_i, \mathbf{w}) = \ell\left(1 - y_i \max_{\mathbf{h}\in\mathcal{H}(\mathbf{x})} (\mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}))\right) \quad (5)$$

where $\Phi_B(\mathbf{x}_i, \mathbf{h}) = \frac{\Phi(\mathbf{x}_i, \mathbf{h})}{\kappa(\mathbf{x}_i)}$. We also add a dummy feature in Φ_B after normalization to adjust the threshold.

2.3. Relation to Other Learning Frameworks

Several discriminative algorithms for learning structured output predictors have been proposed in the literature: these include conditional random fields (Lafferty et al., 2001), max-margin Markov network (Taskar et al., 2004) and structural SVM (Tsochantaridis et al., 2004). These methods use feature vectors of input-output space to capture the interdependency between output variables.

Our framework generalizes other structure learning frameworks. When $B = \emptyset$ and ℓ is the hinge loss or squared hinge loss, it reduces to the structural SVM framework. When $S = \emptyset$ and the goal is the binary task, it is a latent variable framework similar to (Felzenszwalb et al., 2009), which learns a binary SVM over latent structures and is an instantiation of the latent structural SVM of (Yu & Joachims, 2009). Our approach differs from both of these frameworks as it aims to use indirect supervision from one task to help the companion target task.

The objective function Eq. (4) contains two loss terms corresponding to the direct and indirect supervision. While the form of the objective function resembles the objective function of semi-supervised structure learning (Zien et al., 2007; Brefeld & Scheffer, 2006), J-LIS is very different both conceptually and technically. The difference

[2]The function $\kappa(\mathbf{x})$ is not needed when the feature vectors take care of the scaling issue. It is only needed when the $\phi(\mathbf{x}, \mathbf{h})$ is sensitive to the size of $\mathbf{x}$.

stems from our interpretation of the relation between the companion problems allowing us to use invalid examples ($y = -1$), which are not used by semi-supervised learning frameworks. We further discuss the impact of negative examples empirically in Sec. 4.4.

Our work is also conceptually related to Contrastive Estimation (CE) (Smith & Eisner, 2005), where the goal is to learn a structure predictor by pushing the probability mass away from the "bad" neighbors. There are several technical differences between the two approaches. These differences are discussed in Sec. 5 with related experiments.

3. Optimization Algorithm

This section describes the optimization procedure for solving the objective function $Q(\mathbf{w})$ from Eq. (4). First, we study its convexity properties. We can rewrite Eq. (4) as

$$Q(\mathbf{w}) = F(\mathbf{w}) + G(\mathbf{w}),$$

where F and G are given by Eq. (6) and (7), respectively.

$$\frac{\|\mathbf{w}\|^2}{2} + C_1 \sum_{i \in S} L_S(\mathbf{x}_i, \mathbf{h}_i, \mathbf{w}) + C_2 \sum_{i \in B^-} L_B(\mathbf{x}_i, y_i, \mathbf{w}) \quad (6)$$

$$C_2 \sum_{i \in B^+} \ell\left(1 - \max_{\mathbf{h}}(\mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}))\right) \quad (7)$$

If ℓ is convex and non-decreasing, F is convex. However, the function G, which includes a maximization term within it, need not necessarily be convex or concave. This renders the function Q non-convex. This is an effect of the existential quantification over positive examples in Eq. (1).

Since G is not concave, the Concave-Convex procedure (CCCP) (Yuille & Rangarajan, 2003) cannot be applied as in (Yu & Joachims, 2009)[3]. However, we can apply an optimization procedure similar to CCCP *without* requiring G to be concave. We use the fact that our loss function ℓ is non-decreasing, which holds for commonly used loss functions such as hinge loss, squared-hinge loss and logistic loss.

The algorithm 1 iteratively improves the objective function. At the t^{th} iteration of the loop, we denote $\mathbf{w}_t$ to be the current estimation of the weight vector, and denote $\mathbf{h}_i^t = \arg\max_{\mathbf{h}} {\mathbf{w}_t}^T \Phi(\mathbf{x}_i, \mathbf{h})$ to be the best structure for a positive example according to $\mathbf{w}_t$. We then define an approximation function of G using $\mathbf{w}_t$:

$$\hat{G}(\mathbf{w}, \mathbf{w}_t) = G_t(\mathbf{w}) = C_2 \sum_{i \in B^+} \ell(1 - \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}_i^t)). \quad (8)$$

Unlike G, since the $\mathbf{h}_i^t$s are fixed using $\mathbf{w}_t$, the function G_t is convex in $\mathbf{w}$ given ℓ is convex. Now, step 3 of the algorithm minimizes the following *convex* function $A(\mathbf{w}, \mathbf{w}_t)$ to obtain the next estimate of the weights.

$$A(\mathbf{w}, \mathbf{w}_t) = F(\mathbf{w}) + \hat{G}(\mathbf{w}, \mathbf{w}_t) = F(\mathbf{w}) + G_t(\mathbf{w})$$

Algorithm 1 has the following property:

Theorem 1 *If the loss function ℓ is a non-decreasing function, then in Algorithm 1, the objective function (Eq. (4)) will decrease with every iteration. That is, if $\mathbf{w}_t$ is the weight vector from the t^{th} iteration and $\mathbf{w}_{t+1}$ is the weight vector after running the Algorithm 1 for one more iteration. Then, $Q(\mathbf{w}_{t+1}) \leq Q(\mathbf{w}_t)$.*

Proof. For any $\mathbf{h}$ and $\mathbf{w}$, we know that $\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}) \leq \max_{\mathbf{h}'} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}')$. Since ℓ is non-decreasing, $G_t(\mathbf{w}) \geq G(\mathbf{w})$ for any $\mathbf{w}$. From definition of G and G_t, we have $G_t(\mathbf{w}_t) = G(\mathbf{w}_t)$. Thus,

$$\begin{aligned} Q(\mathbf{w}_t) &= F(\mathbf{w}_t) + G(\mathbf{w}_t) = F(\mathbf{w}_t) + G_t(\mathbf{w}_t) \\ &\geq F(\mathbf{w}_{t+1}) + G_t(\mathbf{w}_{t+1}) \\ &\geq F(\mathbf{w}_{t+1}) + G_{t+1}(\mathbf{w}_{t+1}) = Q(\mathbf{w}_{t+1}). \end{aligned}$$

The first inequality is from line 3 in Algorithm 1. □

Algorithm 1 Iterative algorithm for minimizing $Q(\mathbf{w})$ by repeatedly minimizing $A(\mathbf{w}, \mathbf{w}_t)$.

1: Initialize $\mathbf{w}_0$ with direct supervision S.
2: **repeat**
3: $\quad \mathbf{w}_{t+1} \leftarrow \arg\min_{\mathbf{w}} A(\mathbf{w}, \mathbf{w}_t)$
4: **until** convergence
5: Return the final weight vector.

Algorithm 1 minimizes $Q(\mathbf{w})$ by constructing a sequence of convex problems $A(\mathbf{w}, \mathbf{w}_t)$ in each iteration. The algorithm is defined for any convex and non-decreasing loss functions ℓ. Now, we show how the inner loop of the Algorithm 1 (that is, minimizing $A(\mathbf{w}, \mathbf{w}_t)$) can be solved efficiently when $\ell(x) = \max(0, x)^2$, the squared hinge loss.

Our approach adapts the cutting plane strategy of (Joachims et al., 2009) to minimize $A(\mathbf{w}, \mathbf{w}_t)$. We define a working set for each element in S and B^- – let $\mathcal{W}_i$ and $\mathcal{V}_i$ denote the working sets of $\mathbf{x}_i \in S$ and $\mathbf{x}_i \in B^-$ respectively. We minimize A iteratively using Algorithm 2,which first updates the working sets and then solves the following minimization problem :

$$\begin{aligned} \min_{\mathbf{w}} & \frac{1}{2}\|\mathbf{w}\|^2 + C_1 \sum_{i \in S} \xi_i^2 + C_2 \sum_{i \in B} \xi_i^2 \\ \text{s.t. } & \forall i \in S, \mathbf{h} \in \mathcal{W}_i, \xi_i \geq \Delta(\mathbf{h}, \mathbf{h}_i) - \mathbf{w}^T \Phi_{\mathbf{h}_i, \mathbf{h}}(\mathbf{x}_i), \\ & \forall i \in B^-, \mathbf{h} \in \mathcal{V}_i, \xi_i \geq 1 + \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}), \\ & \forall i \in B^+, \xi_i \geq 1 - \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}_i^t) \end{aligned} \quad (9)$$

Note that in the first two sets of constraints above, we have one constraint for each $\mathbf{h}$ in the working set, instead of

[3] CCCP cannot be applied to this split of Q because G is not concave. While it is possible to use other splits, the split proposed in this paper leads to an intuitive and efficient algorithm which has similar guarantees to CCCP.

Algorithm 2 Cutting plane algorithm for optimizing $A(\mathbf{w}, \mathbf{w}_t)$ in Algorithm 1 with square hinge loss.

Require: $\mathbf{w}_t$, the weight vector from t^{th} iteration
$\mathcal{W}_i \leftarrow \emptyset, \forall i \in S$
$\mathcal{V}_i \leftarrow \emptyset, \forall i \in B^-$
$\mathbf{h}_i^t = \arg\max_{\mathbf{h}} \mathbf{w}_t^T \Phi_B(\mathbf{x}_i, \mathbf{h}), \forall i \in B^+$
repeat
  for $i \in B^-$ **do**
    $\mathbf{h}_i^* \leftarrow \arg\max_{\mathbf{h}} \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h})$
    Add $\mathbf{h}_i^*$ to $\mathcal{V}_i$
  end for
  for $i \in S$ **do**
    $\mathbf{h}_i^* \leftarrow \arg\max_{\mathbf{h}} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}) + \Delta(\mathbf{h}_i, \mathbf{h})$
    Add $\mathbf{h}_i^*$ to $\mathcal{W}_i$
  end for
  Update $\mathbf{w}$ by solving Eq. (9)
until no new element is added to any $\mathcal{W}_i$ and $\mathcal{V}_i$
return $\mathbf{w}$

an exponentially large number of constraints, one for each possible $\mathbf{h} \in \mathcal{H}(\mathbf{x})$. Furthermore, using the squared hinge loss ensures that the constraints $\xi_i \geq 0$ are not needed. Hence, the dual formulation of Eq. (9) consists of only box constraints. This allows us to use a very efficient coordinate descent method on the dual formulation of Eq. (9). For brevity, we extend the notation of $\mathcal{V}$ for elements of B^+, where each $\mathcal{V}_i$ is a singleton set consisting of $\mathbf{h}_i^t$.

In the dual of Eq. (9), each variable corresponds to one constraint. We use αs to denote the dual variables. For each $i \in S$, the dual contains $|\mathcal{W}_i|$ variables $\alpha_{i,j}$ for the $\mathbf{h}_{i,j} \in \mathcal{W}_i$. Similarly, we define $\alpha_{i,j}$ for every $\mathbf{h}_{i,j} \in \mathcal{V}_i, i \in B$. The weight vector can be reconstructed from the dual variables using the standard approach. Algorithm 3 summarizes the dual coordinate descent algorithm for minimizing Eq. (9).

Algorithm 3 Dual coordinate descent for minimizing Eq. (9).

repeat
  Pick any variable $\alpha_{i,j}$
  if $i \in S$ **then**
    $\eta_1 = \Delta(\mathbf{h}_i, \mathbf{h}_{i,j}) - \mathbf{w}^T \Phi_{\mathbf{h}_i, \mathbf{h}_{i,j}}(\mathbf{x}_i) + (\sum_{j=1}^{|\mathcal{W}_j|} \alpha_{i,j})/(2C_1)$
    $\eta_2 = \|\Phi_{\mathbf{h}_i, \mathbf{h}_{i,j}}(\mathbf{x}_i)\|^2 + \frac{1}{2C_1}$
  else
    $\eta_1 = 1 - y_i \mathbf{w}^T (\Phi_B(\mathbf{x}_i, \mathbf{h}_{i,j})) + (\sum_{j=1}^{|\mathcal{V}_j|} \alpha_{i,j})/(2C_2)$
    $\eta_2 = \|\Phi_B(\mathbf{x}_i, \mathbf{h}_{i,j})\|^2 + \frac{1}{2C_2}$
  end if
  $\alpha_{i,j} \leftarrow \max(\alpha_{i,j} + \frac{\eta_1}{\eta_2}, 0)$
until convergence

Our optimization algorithm is related to the convex-concave procedure (CCCP) (Yuille & Rangarajan, 2003), which has been used for solving many non-convex optimization problems (Yu & Joachims, 2009). We show that it is not necessary to decompose the objective function into a sum of convex and concave functions to use a CCCP-like iterative procedure. The object recognition work of (Felzenszwalb et al., 2009) uses a similar optimization procedure for their problem. However, not only is the intent of our algorithm is completely different (that work only has labels for the binary problem and the goal is to improve binary classification performance), our optimization algorithm can handle general loss functions.

4. Experiments

We verify the effectiveness of J-LIS by applying it on three NLP tasks defined over complex structures – Phonetic Transliteration Alignment, Information Extraction (McCallum et al., 2000; Grenager et al., 2005) and Part of Speech Tagging (Smith & Eisner, 2005). Our experiments provide insights into the settings in which indirect supervision is most effective [4]. For all our experiments, we selected parameters C_1 and C_2 from the set $\{10^{-1}, 10^0, 10^1\}$ by sampling the data and evaluating the results on a held-out set. We ran our algorithm until the relative change in the objective function became less than 10^{-5}.

4.1. Phonetic Transliteration Alignment

Given a source language named entity (NE) and a corresponding target language NE, the goal of Phonetic Transliteration Alignment is to find the best phonetic alignment between the character sequences of the two NEs.[5] The companion binary classification problem is the task of determining whether two words from different languages correspond to the same underlying entity.

Since a good phonetic alignment is required for two NEs to be considered transliterations of each other, we can use this information to guide the alignment predictor. Given an input example (in our case, an English and Hebrew NE pair), the best sequence alignment can be efficiently found using a dynamic programming algorithm. Our feature vector corresponds to the alignment edges, using the corresponding character sequences. The normalization term $\kappa(\mathbf{x})$ is set to the number of characters in the Hebrew NE.

We used the English-Hebrew data-set from (Chang et al., 2009), consisting of 300 pairs and manually annotated the alignments between the NEs' segments. We used 100 pairs for training and 200 for testing. Our binary data, obtained by crawling the web, consists of 1,000 positive and 10,000 negative pairs[6]. Note that obtaining the binary supervision is considerably easier than labeling the alignment.

[4] Further details will be provided at `http://L2R.cs.uiuc.edu/~cogcomp/wpt.php?pr_key=INDIRECT`

[5] The character sequences can consist of multiple characters.

[6] Negative pairs were created by pairing an English NE to a random Hebrew NE.

Table 1. **F1 measure for the phonetic transliteration alignment task.** The amount of direct supervision used for the structure prediction task ($|S|$) varies across the rows of the table, while the size of indirect supervision ($|B|$) changes across the columns. The first column, ($|B| = 0$) is the standard structural SVM (SSVM). Results show that indirect supervision is especially effective when little supervision exists for the structure task. The error reduction compared to structural SVM is in parentheses.

	Size of B			
$\|S\|$	0 (SSVM)	$2k$	$4k$	$8k$
10	72.9	78.8	79.8	80.0 (26.2%)
20	82.1	84.6	84.7	85.4 (18.4%)
40	85.7	86.9	87.2	87.4 (12.0%)
80	88.6	89.4	89.0	89.4 (7.1%)

To measure the impact of indirect supervision, we vary the size of the direct and indirect supervision sets. We report the F1 measure for the alignment in Table 1. Adding indirect supervision improves the structure predictor significantly. For example, when we have only ten structured labeled pairs, the error reduction rate is 26%.

4.2. Part-Of-Speech tagging

Our POS data is from the Wall Street Journal corpus (Marcus et al.); we used 25600 tokens for training and testing (which correspond to 1000 sentences). Using a separate set of sentences corresponding to 25600 tokens, we generated 2000 indirect supervision examples. The 1000 negative examples among these were generated by randomly shuffling sentences in the corpus. To allow for more experiments, we adopted the coarse tags from (Smith & Eisner, 2005) here. We did not use a tagging dictionary, and all tags were considered for every word. Our model was a first order Markov model with spelling features representing capitalization and suffixes of the current word. We set our $\kappa(\mathbf{x})$ to be the number of words in the sentence $\mathbf{x}$.

The results of the experiments are summarized in Fig. 2. We observe that indirect supervision is most effective when the size of the structured labeled data-set is small. For example, when S consists of 200 tokens, adding indirect supervision improves the predictor from 71% to 76%. As in the transliteration domain, increasing the indirect supervision often results in better performance, the trend is very clear in Fig. 2. While the effect of indirect supervision decreases when the supervision set for the structure learning problem increases, we found that J-LIS is competitive with structural SVM even when we used all the labeled data (25.6k). The supervised SVM achieves 93.66% with all labeled examples, compared to J-LIS 's 93.72%.

4.3. Information Extraction

Information Extraction (IE) is the task of identifying predefined fields in text. We report results for two IE tasks:

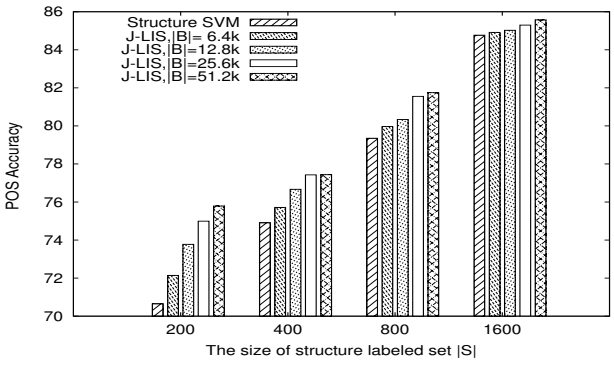

Figure 2. **Results for the part-of-speech tagging.** Adding indirect supervision significantly improves the results. Also, the results are better when more indirect supervision is used. We report the size of the data-sets used by counting the total number of **tokens** due to the variance in sentences size. See the text for more details. Even when *all* of our labeled data is used (25.6k), SSVM and J-LIS are comparable (SSVM: 93.66% and J-LIS :93.72%).

(i) Extracting fields from citation (e.g., author, title, year) (McCallum et al., 2000), and (ii) Extracting fields from advertisements (e.g., size, rent and neighborhood) (Grenager et al., 2005). The companion binary problem is to classify whether a text is a well structured citation/advertisement. For citations, we used 300 structured labeled examples for training, 100 for development and 100 for testing. In the advertisements domain, we used 100 labeled examples for training, development and testing. For each domain, our positive data contains 1k entries (50k tokens for the citation domain and 200k tokens for the advertisement domain). We generate 1k negative entries for each domain by randomly shuffling the tokens. We use features corresponding to the current word and previous state allowing us to use Viterbi to find the best sequence efficiently. The $\kappa(\mathbf{x})$ is set to the number of tokens of this entry.

The token-level accuracy results for both domains are summarized in Table 2, where we vary $|S|$ and fix $|B|$ for each domain. In the advertisement domain, when the number of tokens in S is 500 tokens, structural SVM attains an accuracy of approximately 58%. Adding indirect supervision boosts the accuracy to over 66%, which even outperforms the structural SVM results with $|S| = 1000$. In the citation domain, we observe similar trends.

4.4. The importance of negative examples

The key difference between J-LIS and previous work on discriminative semi-supervised structured output prediction (e.g., (Brefeld & Scheffer, 2006; Zien et al., 2007)) is the use of *invalid* ($y = -1$) examples. These approaches use well-formed unlabeled examples for training. These correspond only to our $y = +1$ class. To provide further insight into the role of negative examples, we isolate the

Table 2. **Results for two IE tasks.** $|S|$ (the structured supervised set) is measured by the number of **tokens**. Performance is evaluated by token-level accuracy with a fixed indirect supervision set B. The results are bold faced when the improvement obtained by J-LIS is statistically significantly under paired-t test $p < 0.01$.

Advertisements			Citations		
$\|S\|$	**SSVM**	J-LIS	$\|S\|$	**SSVM**	J-LIS
500	57.18	**66.60**	100	58.24	**64.05**
1000	64.93	**70.09**	400	70.53	**73.63**
2000	69.57	**72.75**	1600	82.39	**84.28**
4000	74.24	**75.80**	6400	90.15	90.33
19k	78.07	79.00	9k	92.24	92.31

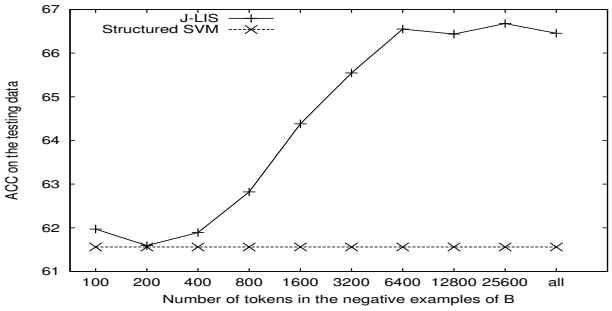

Figure 3. Impact of negative examples in the citation domain. Results show that as more negative examples ($y = -1$) are used, indirect supervision improves performance.

contribution of the invalid examples in the indirect supervision dataset by fixing the number of positive examples, and show the effect of varying the number of negative examples in the citation domain. We fix the structured labeled set ($|H| = 100$ tokens). The binary indirect training set is created as follows – positive examples are fixed (34767 tokens) and the number of negative examples is varied.

Results (Fig. 3) show that increasing the number of negative examples improves the performance of the structure predictor. This stresses the advantage of J-LIS over the standard discriminative semi-supervised approaches which cannot gain from negative examples.

5. Discussion: Focus on Y

When the size of S is zero CE (Smith & Eisner, 2005) is conceptually related to this work. However, the goal of J-LIS is to *jointly* learn from both structured and binary supervision, while CE is not designed to use labeled structures. Therefore, we compare to CE by restricting J-LIS to use $|S| = 0$. Next, we describe the conceptual difference between the approaches and then empirically compare the two algorithms and Expectation-Maximization(EM).

Even without any labeled structure, J-LIS is less restricted than CE. First, in CE, a "good" example and its "bad" neighbors need to be grouped together and CE cannot be directly applied when the relationship between good and bad examples is not known. In contrast, our framework can be directly applied to *existing* binary datasets. Moreover, CE needs to marginalize over all possible hidden structures, while J-LIS only looks for the best structure. Hence, the practical computational cost of the inference problem is lower. This property also allows us to incorporate complex domain specific constraints, which as previous work has shown can significantly boost the performance of structure predictors (Roth & Yih, 2005; 2007; Martins et al., 2009). It is not clear how to use arbitrary constraints in CE without using an approximated inference procedure.

Following (Smith & Eisner, 2005), we adopt the commonly used tagging dictionary assumption: for any word, we know all its possible POS (e.g. 'play' can be a verb or a noun). We used a 96K word subset of the WSJ corpus, and evaluated on ambiguous words (that is, words with more than one allowed tag), as in the CE work. All models used the second order Markov assumption and exactly the same features. For J-LIS, we generate four negative examples for each positive one by randomly shuffling its words. We compare to the closest CE experimental setting which transposes neighboring words.

The token accuracy of EM and CE are 60.9% and 74.7%, respectively in a pure unsupervised learning. When hyperparameters are tuned using supervised data, the performance can go up to 62.6% and 79.0%. The restricted J-LIS attains 70.1%, which is significantly better than EM but worse than CE. We hypothesize that the reason is that J-LIS only finds a single best structure. Without any initialization, it is likely that many structures are assigned the same score as the best one[7]. While this does not affect CE, which sums over the score of all structures, J-LIS might commit to a solution too early. We verified this intuition by adding merely 5 structured labeled sentences that provide better initial point, resulting in an accuracy of 79.1%.

Using structure as indirect supervision In this section we consider the reverse question: can structured data act as indirect supervision for the binary task. Since J-LIS does not make any assumptions about S and B, it can be applied *directly* when the binary task is the target. We briefly describe experimental results in this setting using the *transliteration identification* task (determining if a given NE pair is a transliteration pair). This is the companion problem to the transliteration alignment problem considered in Sec. 4.1. We matched each English NE in the test data with the best Hebrew NE using the classifier's confidence and measured the accuracy of the top ranking prediction. Our test data consists of 200 English and Hebrew NEs.

[7]It is therefore important for J-LIS to have an inference algorithm capable of randomly choosing among equivalent solutions.

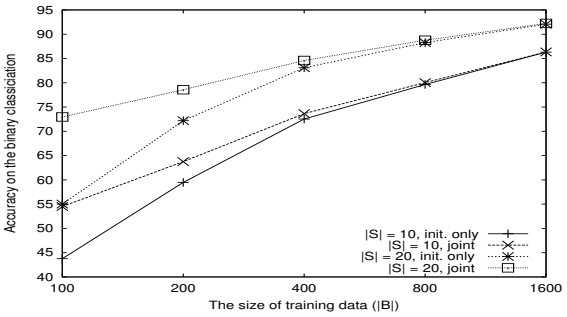

Figure 4. The impact of structured labeled data when binary classification is our target. Results (for transliteration identification) show that joint training significantly improves performance, especially when direct supervision is scarce.

There are two ways of using S to improve binary prediction: (1) Initialize the weight vector using S, and apply J-LIS to B. (2) Initialize the weight vector with S, and then apply J-LIS on both B and S (the **joint approach**). We vary the size of the $|B|$ from 100 to 800 and keep the positive to negative ratio to 0.1 and report results for $|S| \in \{10, 20\}$. Fig. 4 shows that increasing $|S|$ improves accuracy. Furthermore, the joint method performs significantly better than using S for initialization only.

6. Conclusion

This work studies two companion problems – structured output prediction and a binary decision problem over the structure. The key contribution of this work is the development of a discriminative joint learning framework, J-LIS, that exploits the relationship between the two problems. Consequently it can make use of easy-to-get supervision for the binary problem to improve structure prediction, where supervision is hard to get. We apply our framework to three structure learning tasks - phonetic transliteration alignment, information extraction and part-of-speech tagging and show significant empirical improvements. Interestingly, in all three domains the most significant improvement is obtained when little direct supervision is available for the structure prediction task, thus demonstrating the benefit of using our framework especially in data-poor domains, where more supervision is required.

Acknowledgment We thank Derek Hoiem and James Clarke for their insightful comments. This research was partly sponsored by the Army Research Laboratory (ARL) (accomplished under Cooperative Agreement Number W911NF-09-2-0053) and by Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the ARL or AFRL.

References

Brefeld, U. and Scheffer, T. Semi-supervised learning for structured output variables. In *ICML*, 2006.

Chang, M., Goldwasser, D., Roth, D., and Tu, Y. Unsupervised constraint driven learning for transliteration discovery. In *NAACL*, 2009.

Chang, M., Goldwasser, D., Roth, D., and Srikumar, V. Discriminative learning over constrained latent representations. In *NAACL*, 2010.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 2009.

Grenager, T., Klein, D., and Manning, C. Unsupervised learning of field segmentation models for information extraction. In *ACL*, 2005.

Joachims, T., Finley, T., and Yu, Chun-Nam. Cutting-plane training of structural svms. *Machine Learning*, 2009.

Klementiev, A. and Roth, D. Named entity transliteration and discovery in multilingual corpora. In Goutte, Cyril, Cancedda, Nicola, Dymetman, Marc, and Foster, George (eds.), *Learning Machine Translation*. 2008.

Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*.

Martins, A. F. T., Smith, N. A., and Xing, E. P. Polyhedral outer approximations with application to natural language parsing. In *ICML*, 2009.

McCallum, A., Freitag, D., and Pereira, F. Maximum entropy markov models for information extraction and segmentation. In *ICML*, 2000.

Roth, D. and Yih, W. Integer linear programming inference for conditional random fields. In *ICML*, 2005.

Roth, D. and Yih, W. Global inference for entity and relation identification via a linear programming formulation. In *Introduction to Statistical Relational Learning*, 2007.

Smith, N. and Eisner, J. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL*, 2005.

Taskar, B., Guestrin, C., and Koller, D. Max-margin markov networks. In *NIPS*, 2004.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

Yu, C. and Joachims, T. Learning structural svms with latent variables. In *ICML*, 2009.

Yuille, A. L. and Rangarajan, A. The concave-convex procedure. *Neural Computation*, 2003.

Zien, A., Brefeld, U., and Scheffer, T. Transductive support vector machines for structured variables. In *ICML*, 2007.