Mach Learn (2012) 88:399–431
DOI 10.1007/s10994-012-5296-5

Structured learning with constrained conditional models

Ming-Wei Chang · Lev Ratinov · Dan Roth

Received: 6 May 2008 / Accepted: 6 May 2012 / Published online: 7 June 2012

Abstract Making complex decisions in real world problems often involves assigning values to sets of interdependent variables where an expressive dependency structure among these can influence, or even dictate, what assignments are possible. Commonly used models typically ignore expressive dependencies since the traditional way of incorporating non-local dependencies is inefficient and hence leads to expensive training and inference.

The contribution of this paper is two-fold. First, this paper presents Constrained Conditional Models (CCMs), a framework that augments linear models with declarative constraints as a way to support decisions in an expressive output space while maintaining modularity and tractability of training. The paper develops, analyzes and compares novel algorithms for CCMs based on Hidden Markov Models and Structured Perceptron. The proposed CCM framework is also compared to task-tailored models, such as semi-CRFs.

Second, we propose CoDL, a constraint-driven learning algorithm, which makes use of constraints to guide semi-supervised learning. We provide theoretical justification for CoDL along with empirical results which show the advantage of using declarative constraints in the context of semi-supervised training of probabilistic models.

Keywords Semi-supervised learning · Information extraction · Natural language processing

1 Introduction

Decision making in domains such as natural language processing is characterized by ambiguity and partial or imperfect information sources, which necessitate the use of models

Editor: Hal Daume III.

M.-W. Chang · L. Ratinov (✉) · D. Roth
Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, IL, USA
e-mail: ratinov2@illinois.edu

M.-W. Chang
e-mail: mchang21@illinois.edu

D. Roth
e-mail: danr@illinois.edu

learned from data. Decisions made with these models often involve assigning values to *sets* of interdependent variables where the expressive dependency structure among variables of interest can influence, or even dictate, what assignments are possible. To cope with these difficulties, problems are typically modeled as stochastic processes involving both output variables (those whose values are sought) and the information sources, often referred to as input or observed variables.

The dominant family of models used in these tasks are *linear models*, which can be represented as a weight vector $\mathbf{w}$, corresponding to a set of feature functions $\{\Phi\}$. For an input instance $\mathbf{x}$ and an output assignment $\mathbf{y}$, the "score" of the instance can be expressed as a weighted sum of feature functions: $f(\mathbf{x}, \mathbf{y}) = \sum w_i \phi_i(\mathbf{x}, \mathbf{y})$. When the model is evaluated on an unlabeled instance $\mathbf{x}$, the aim is to *infer* the best assignment to the output variables,

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum w_i \phi_i(\mathbf{x}, \mathbf{y}). \tag{1}$$

We refer to this problem as the "inference problem". Many different discriminative and generative learning algorithms can be represented as linear models. This view has given rise to developing learning algorithms for structured models expressed linearly over more expressive feature functions (Roth 1999; Collins 2002; Lafferty et al. 2001).

While linear models share the same "prediction function" (Eq. (1)), there exist several fundamentally different *learning algorithms* for these models. One approach is to completely ignore the output structure at the learning stage (by learning local models that make independent local decisions), while enforcing coherent assignments only at the inference stage (Roth and Yih 2004, 2007). Another learning solution is to, directly or indirectly, model the dependencies among the output variables in the learning process and thus induce models that optimize a global performance measure. In this scenario, to allow efficient training and inference, the model of the joint distribution is factored into functions of subsets of the variables, yielding models such as Markov Random Fields (MRFs), Conditional Random Fields (CRFs) and Hidden Markov Models (HMMs). Although, in general, the feature functions $\Phi(\mathbf{x}, \mathbf{y})$ used in a linear representation such as in Eq. (1) can represent any function of $\mathbf{x}$ and $\mathbf{y}$, it is typical to use $\Phi(\mathbf{x}, \mathbf{y})$ which only encode *local relationships*, as in the linear representation of first/second-order HMMs (Roth 1999; Collins 2002; Lafferty et al. 2001). This makes the process of finding the best assignment given an instance $\mathbf{x}$ tractable. However, such restrictions usually render the feature functions not expressive enough to capture non-local dependencies that are present in the problem.

In many problems, dependencies among output variables have non-local nature, and incorporating them into the model as if they were probabilistic phenomena can undo a great deal of the benefit gained by the aforementioned factorization, as well as making the model more difficult to design and understand. For example, consider an information extraction task where two particular types of entities cannot appear together in the same document. Modeling mutual exclusion in the scenario where n random variables can be assigned mutually exclusive values introduces n^2 pairwise edges in the graphical model, with obvious impact on training and inference. Obviously, this is very expensive given that a lot of parameters are being wasted in order to learn something the model designer already knows. For example, in order to capture such constraints by higher order HMMs or CRFs, we need to build a T-order model which can consider all connections if there are T tokens in $\mathbf{x}$. This requires a significant increase in the number of parameters even though we actually know that all the weights on the links between y_i and y_j should be $-\infty$ if y_i equals to y_j. In short, HMMs and CRFs do not have a way to encode the knowledge *directly* but only *indirectly*, by adding more features or increasing the order of the models (Roth and Yih 2005).

However, inference problems in high-order models are very expensive and achieving good performance by learning a more complex model requires more labeled examples. Therefore, high order models will have a huge disadvantage when the number of examples is limited. In short, non-local and first-order relationships can be very difficult to model using only *local* features and might require a lot of training examples to achieve good results.

In this paper, we address the need of having a general framework that allows one to encode expressive knowledge about the model *directly* and develop a general learning framework to address this issue. The contributions of the paper are as follows:

1. We propose the **Constrained Conditional Model (CCM)** framework, which provides a *direct* way to inject prior knowledge into a conditional model, in the form of **constraints**. One advantage of CCMs is that it allows combining simple models with *declarative and expressive* constraints. This is an effective approach to making probabilistic models expressive. Therefore, CCMs can be considered as an *interface* for incorporating knowledge into off-the-shelf statistical models without designing a task-specific model. Note that adding constraints to CCMs does not enlarge the feature space but rather augments the simple linear model. Along with appropriate training approaches that we discuss later, we need to learn simpler model than standard high order probabilistic models but can still make decisions with expressive models. Since within CCMs we combine declarative constraints, possibly written as first order logic expressions (Rizzolo and Roth 2007), with learned probabilistic models, we can treat CCMs as a way to combine or bridge logical, declarative, expressions and learning statistical models. We also discuss how to solve inference problems with expressive constraints efficiently in Sect. 2.2.
2. Based on the principle introduced by CCMs, we introduce **HMM$^{\mathbf{CCM}}$**, a constraint-infused Hidden Markov Model. We demonstrate how to train and test HMM$^{\mathrm{CCM}}$ in a principled way and show that adding little knowledge can improve the model significantly.
   Note that by modeling the constraints directly, the inference problem in Eq. (1), becomes harder to solve, compared to the one used by low order HMMs/CRFs. As we show later, such a sacrifice is usually very rewarding in terms of final performance. Moreover, we show that constraints do not add any overhead to our *learning* algorithm of HMM$^{\mathrm{CCM}}$ under our assumption.
3. We show that prior knowledge plays a crucial role when the amount of labeled data is limited. We empirically show that incorporating high-level knowledge via CCMs significantly improves the results of both *supervised learning* and *semi-supervised learning*.
   Note that semi-supervised learning results are especially interesting, since we can consider constraints as a supervision resource that guides the semi-supervised learning procedure.

This paper formally defines CCMs so that it is easier to apply constraints to statistical models in both supervised and semi-supervised settings. Moreover, we provide a principled justification for the algorithms proposed in (Chang et al. 2007) (with modifications) and obtain better empirical results. Finally, this paper includes a wide set of experiments that show the properties of the HMM$^{\mathrm{CCM}}$ algorithm and compares it to other algorithms.

Note that we are not the first to point out the importance of long distance relationships and other approximate supervised training algorithms have been proposed (see Sect. 7 for more details). However, we want to stress that in CCMs, the notion of constraints is different and more general. For example, the CCM framework offers the possibility to separate models (features) and constraints. Therefore, it is possible to apply constraints to a trained model *directly* without re-training the model. Moreover, such separation is the key to the success

of our semi-supervised learning algorithm, which uses constraints as a form of supervision. We clarify this point later in the text.

The rest of the paper is organized as follows: Sect. 2 formally defines Constrained Conditional Models. We introduce an instance of CCMs based on a Hidden Markov Model in Sect. 3. In Sect. 4 we introduce the tasks and the data on which the algorithms will be tested. The experimental results are presented in Sect. 5. In Sect. 6 we discuss other options of using a CCM, beyond a Hidden Model, and provide some results on learning CCMs with structured perceptron. We discuss related work in Sect. 7 and make conclusions in Sect. 8.

2 Constrained conditional model

CCMs target structured prediction problems. Given a point $\mathbf{x}$ in an input space $\mathcal{X}$, the goal is to find a labeled assignment $\mathbf{y}$ in the set of all possible output structures for $\mathbf{x}$, $\mathcal{Y}(\mathbf{x})$. For example, in part-of-speech (POS) tagging, $\mathcal{Y}(\mathbf{x})$ is the set of all possible POS tags for a given input sentence $\mathbf{x}$.

Given a set of feature functions $\Phi = \{\phi_i(\cdot)\}_{i=1}^n$, $\phi_i : \mathcal{X} \times \mathcal{Y} \to R$, which typically encode the *local* properties of a pair $(\mathbf{x}, \mathbf{y})$ (often, the image of ϕ_i is $\{0, 1\}$), the "score" of a structure $\mathbf{y}$ of a linear model can be represented as

$$f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{x}, \mathbf{y}).$$

The prediction function of this linear model is $\arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} f(\mathbf{x}, \mathbf{y})$.

Constrained Conditional Models provide a general **interface** that allows users to easily combine domain knowledge (which is provided by humans) and statistical models (which are learned from the data). In this paper, we represent domain knowledge as a (usually small) set of constraints $\mathcal{C} = \{C_k(\cdot)\}_{k=1}^m$, $C_k : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ which encode predicates over a pair $(\mathbf{x}, \mathbf{y})$. If $C_k(\mathbf{x}, \mathbf{y}) = 1$, it means that the pair $(\mathbf{x}, \mathbf{y})$ violates the constraint C_k. For each constraint, we are also provided a function $d_{C_k} : \mathcal{X} \times \mathcal{Y} \to R$ that measures the degree to which the constraint C_k is violated in a pair $(\mathbf{x}, \mathbf{y})$. While there are different ways to estimate d_{C_k}, in this paper, we define the "violation function" as follows. Let

$$\mathbf{y}_{[1\ldots i]} = (y_1, y_2, \ldots, y_i),$$

be a partial assignment of $\mathbf{y}$. Then

$$d_{C_k}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{|y|} \hat{C}_k(\mathbf{x}; \mathbf{y}_{[1\ldots i]}), \tag{2}$$

where $\hat{C}_k(\mathbf{x}; \mathbf{y}_{[1\ldots i]})$ is a binary function which indicates whether y_i violates the constraint C_k with respect to a partial assignment $\mathbf{y}_{[1\ldots i-1]}$. Note that for some constraints, the violation cannot be calculated with partial assignments. In these cases, $\hat{C}_k$ will return 0 to indicate the constraints i is not violated according to the current partial assignment.

A **Constrained Conditional Model** can be represented using two weight vectors: the feature weight vector $\mathbf{w}$ and the constraint penalty vector ρ. The score of an assignment

$\mathbf{y} \in \mathcal{Y}$ for an instance $\mathbf{x} \in \mathcal{X}$ can then be obtained by[1]

$$f_{\Phi,C}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{x}, \mathbf{y}) - \sum_{k=1}^{m} \rho_k d_{C_k}(\mathbf{x}, \mathbf{y}). \tag{3}$$

A CCM then selects the best structure using the inference problem

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} f_{\Phi,C}(\mathbf{x}, \mathbf{y}), \tag{4}$$

as its prediction.

Note that Eq. (3) allows using both "hard constraints" (constraints that should not be violated) and "soft constraints" (constraints that can occasionally be violated). Assume that the constraint set can be partitioned into a soft constraint set $\mathcal{S}$ and a hard constraint set $\mathcal{H}$ ($\mathcal{H} \cap \mathcal{S} = \emptyset$ and $\mathcal{H} \cup \mathcal{S} = \mathcal{C}$). The set of "feasible" structures for a given input $\mathbf{x}$ is then reduced to

$$\bar{\mathcal{Y}}(\mathbf{x}) = \left\{ \mathbf{u} \mid \mathbf{u} \in \mathcal{Y}(\mathbf{x}), C_k(\mathbf{x}, \mathbf{u}) = 0, \forall C_k \in \mathcal{H} \right\}$$

Eq. (4) can be rewritten as

$$\arg\max_{\mathbf{y} \in \bar{\mathcal{Y}}(\mathbf{x})} \sum_{i=1}^{n} w_i \phi_i(\mathbf{x}, \mathbf{y}) - \sum_{k: C_k \in S} \rho_k d_{C_k}(\mathbf{x}, \mathbf{y}).$$

Note that a CCM is not restricted to be trained with any particular learning algorithm. The key goal of a CCM is to allow combining constraints and models in the test phase. Similarly to other linear models, specialized algorithms may need to be developed to train CCMs. Notice also that the left component in Eq. (3) may stand for multiple linear models, trained separately. Unlike standard linear models, we assume the availability of some prior knowledge, encoded in the form of *constraints*. When there is no prior knowledge, there is no difference between CCMs and other linear models.

2.1 The benefits of distinguishing between constraints and features

In Eq. (3), the constraints term (the second term) appears to be similar to the features term (the first time). In fact, the decision of whether to use constraints or features to express long distance relationships can sometimes be a design choice. However, it is important to note that both in this work and in many other recent publications (Roth and Yih 2004, 2005; Chang et al. 2007; Graca et al. 2007; Bellare et al. 2009; Carlson et al. 2010; Ganchev et al. 2010), people have demonstrated the importance of separating features and constraints. In this section we discuss this issue in details.

We note that the distinction is neither obvious nor natural. For example, it is sometimes possible to clamp the weights in CRFs/MRFs to achieve the "constraints behavior" and our encoding of constraints with FOL-like expressions can sometimes be seen as nothing more than a syntactic sugaring. However, consider the constraint "two labels A and B cannot appear in the same assignment". Adding the $O(|\mathbf{y}|^2)$ weights and clamping them requires a special machinery, for which our declarative formulation seems extremely appropriate.

[1] Recall that n is the number of features and is typically very large, and m is the number of constraints, typically small.

– Hard constraints vs. features:
  While we simplified our notation in Eq. (3), the constraints term is different from the features term because it can be used to enforce hard constraints. Hence, it is necessary to separate constraints and features.
– Reusing and improving existing models with expressive constraints:
  It is often expensive to retrain a complex NLP system. While choosing features or constraints to express long distance relationships can be a design choice, adding more features often requires expensive retraining. Moreover, in Roth and Yih (2004), it is proposed to use constraints to *combine* two independently trained models. Note that if we model the long distance constraints as features, we need to train these two models *jointly*, which can be significantly more expensive than training them separately by separating constraints from the features.
  The benefit of adding constraints to existing models without retraining is partly due to the fact that constraints can be a lot more expressive than the features used in the existing models. Note that the predicate $C(\mathbf{x}, \mathbf{y})$ should be thought of as similar to a "first order logic expression", which is very different from features $\Phi(\mathbf{x}, \mathbf{y})$. An example of $C(\mathbf{x}, \mathbf{y})$ might be "1, if all y_is in the sequence y are assigned different values, 0 otherwise", which is very difficult to model using features. We note that usually, due to their first order logic functionality, the set of constraints is compact. In fact, in our experiments, we only have about 10 constraints. Compared to the feature vector, which may contain thousands of features, due to their propositional "grounded" nature, the size of $C(\mathbf{x}, \mathbf{y})$ is quite small. Moreover, $C(\mathbf{x}, \mathbf{y})$ usually encodes long distance relationships among $\mathbf{y}$ variables, which cannot be captured by the feature functions $\Phi(\mathbf{x}, \mathbf{y})$.
– Implications on learning algorithms:
  Distinguishing expressive constraints from models also impacts the learning performance. Many recent works have shown the benefits of keeping the existing model and treating the expressive constraints as a form of supervision (Chang et al. 2007; Graca et al. 2007; Bellare et al. 2009; Carlson et al. 2010; Ganchev et al. 2010). As we show in this work, using constraints as a supervision resource can be very effective when there are few labeled examples, e.g., in a semi-supervised setting.
  In a supervised setting, we distinguish the constraints from features in Eq. (3) because the constraints should be trusted most of the time. Therefore, the penalties ρ can be fixed or handled separately. For example, if we are confident about our knowledge, rather than learning the $\{\rho_j\}$, we can directly set them to ∞, thus forcing the chosen assignment $\mathbf{y}$ to satisfy the constraints. These issues are discussed in details later in the paper.
– Efficiency:
  Another difference between ρ and $\mathbf{w}$ is that ρ should always be positive. The reason is that $d_{C_i}(\mathbf{x}, \mathbf{y}) \geq 0$ and the assignments that violate the constraints should be penalized (see Eq. (3)). This allows us to design an admissible heuristic and speed up exact inference using A^* search. This nice result hinges on distinguishing constraints from features. This is of particular importance, since the constraints could be non-local, therefore efficient dynamic programming algorithms are not applicable.

There are several additional advantages of using constraints. First, constraints provide a platform for encoding prior knowledge, possibly expressed as high level predicates. As we will show later, this is especially important when the number of labeled instances is small. Second, constraints can be significantly more expressive than features commonly used by linear models. Third, adding constraints can *simplify* the modeling of a complex structured output problem. Instead of building a model from complex features, with the additional training cost this implies, CCMs provide a way to combine "simple" learned models with

a small set of "expressive" constraints to support final decisions. Importantly, combining simple models with constraints often results in better performance. For example, the top-ranking system in the CoNLL 2005 shared task uses a CCM approach and outperforms many systems built using complex models (Punyakanok et al. 2005a). There is a lot of more recent literature that provides additional evidence for this.

2.2 Inference with constraints

Adding expressive constraints comes with a cost—the dynamic programming inference algorithms often used in off-the-shelf statistical models can no longer be applied. In this section, we discuss three different types of inference algorithms that allow solving the inference problem in Eq. (4) with expressive constraints.

2.2.1 Integer linear programming

In the earlier related works that made use of constraints, the constraints were assumed to be Boolean functions; in most cases, a high level (first order logic) description of the constraints was compiled into a set of linear inequalities, and exact inference was done using an integer linear programming formulation (ILP) (Roth and Yih 2004, 2005, 2007; Punyakanok et al. 2005a; Barzilay and Lapata 2006; Clarke and Lapata 2006). Although ILP can be intractable for very large-scale problems, it has been shown to be quite successful in practice when applied to many practical NLP tasks (Roth and Yih 2005, 2007).

2.2.2 A^* search

Recall that the inference problem for CCMs is define by (as in Eq. (4)):

$$\max_{\mathbf{y}} f_{\Phi,C}(\mathbf{x},\mathbf{y}) = \max_{\mathbf{y}} \mathbf{w}^T \Phi(\mathbf{x},\mathbf{y}) - \sum_{k=1}^{m} \rho_k d_{C_k(\mathbf{x},\mathbf{y})}.$$

Assume that there exists an efficient dynamic programming algorithm that computes $\arg\max \mathbf{w}^T \Phi(\mathbf{x},\mathbf{y})$ without considering the constraints.[2] This implies that if we ignore the constraints, given a partial label assignment $\mathbf{y}_{[1\ldots i]}$, we can efficiently complete the label assignment $\mathbf{y}_{[(i+1)\ldots|y|]}$ without considering the constraint penalty, where $|y|$ represents the total number of "parts" of the output structure. That is, we can solve the following optimization problem efficiently and exactly:

$$h(\mathbf{x},\mathbf{y}_{[1\ldots i]}) = \max_{\mathbf{y}_{[(i+1)\ldots|y|]}} \mathbf{w}^T \Phi(\mathbf{x},\mathbf{y}_{[(i+1)\ldots|y|]} \mid \mathbf{y}_{[1\ldots i]}) \tag{5}$$

Note that in the above equation, $\mathbf{y}_{[1\ldots i]}$ is fixed and we search over the rest of an assignment $\mathbf{y}_{[(i+1)\ldots|y|]}$ to complete $\mathbf{y} = \mathbf{y}_{[1\ldots i]} \cdot \mathbf{y}_{[(i+1)\ldots|y|]}$. The value $\mathbf{w}^T \Phi(\mathbf{x},\mathbf{y}_{[(i+1)\ldots|y|]} \mid \mathbf{y}_{[1\ldots i]})$ is the partial score for the $\mathbf{y}_{[(i+1)\ldots|y|]}$ with the given prefix and hence,

$$\mathbf{w}^T \Phi(\mathbf{x},\mathbf{y}_{[1\ldots i]} \cdot \mathbf{y}_{[(i+1)\ldots|y|]}) = \mathbf{w}^T \Phi(\mathbf{x},\mathbf{y}_{[1\ldots i]}) + \mathbf{w}^T \Phi(\mathbf{x},\mathbf{y}_{[(i+1)\ldots|y|]} \mid \mathbf{y}_{[1\ldots i]}).$$

[2]This is the case for virtually all off-the-shelf structured statistical models, since their feature function $\Phi(\mathbf{x},\mathbf{y})$ can be decomposed. For example, if the task is a sequential tagging task and the feature function only captures the relationship of consecutive tokens, there exists an efficient Viterbi algorithm that can return the optimal sequence.

We can perform this factorization because of the assumption that the feature function can be decomposed.

We also define g as the function that returns the score (including constraint penalties) of the current partial assignment $\mathbf{y}_{[1\ldots i]}$:

$$g(\mathbf{x}, \mathbf{y}_{[1\ldots i]}) = \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}_{[1\ldots i]}) - \sum_{j=1}^{m} \rho_j d_{C_j}(\mathbf{x}, \mathbf{y}_{[1\ldots i]}). \tag{6}$$

Next, we show that using g and h, the A^*algorithm can always return the optimal solution of the CCM inference problem.

Theorem 1 *Assume that $\rho_k \geq 0$ for $k = 1 \ldots m$ (that is, we always penalize the assignment that violates the constraints) and that the A^* algorithm uses $h(\mathbf{x}, \mathbf{y}_{[1\ldots i]})$ (Eq. (5)) as the heuristic function and uses $g(\mathbf{x}, \mathbf{y}_{[1\ldots i]})$ (Eq. (6)) to obtain the score of the current partial assignment as an estimation of the final score (that is, we use $g(\mathbf{x}, \mathbf{y}_{[1\ldots i]}) + h(\mathbf{x}, \mathbf{y}_{[1\ldots i]})$). Then, the A^* algorithm will always return the optimal solution of Eq. (3), the CCM inference problem.*

Proof The proof follows by showing that h is an admissible heuristic function. Since $\rho_k \geq 0$, for $k = 1 \ldots m$ and by the definition of Eq. (3),

$$\max_{\mathbf{u}, \mathbf{u}_{[1\ldots i]} = \mathbf{y}_{[1\ldots i]}} f_{\Phi, C}(\mathbf{x}, \mathbf{u}) \leq g(\mathbf{x}, \mathbf{y}_{[1\ldots i]}) + h(\mathbf{x}, \mathbf{y}_{[1\ldots i]}).$$

Hence, we never underestimate the final score given the current partial assignment $\mathbf{y}_{[1\ldots i]}$. Given that we are solving a maximization problem, h is an admissible heuristic function for the A^*algorithm. □

2.2.3 Approximate search

While the A^* algorithm is technically sound, in this paper, we use beam search to approximate the solution for the inference problem in Eq. (4). The advantage of using this procedure is that the memory usage of beam search is fixed while the memory usage of the A^* algorithm can be potentially large. We found that the approximate inference procedure performs very well in our experiments. The comparison of the three proposed inference algorithms on other domains is an interesting issue to address in future research.

3 Learning constrained conditional models based on HMM

In this section, we demonstrate how to apply the idea of CCMs to a commonly used Hidden Markov Model (HMM) and propose **HMM$^{\mathbf{CCM}}$**. The new model naturally incorporates the constraints into an HMM and makes it a very powerful model. In Sect. 2.2, we showed that while the constraints introduce some overhead to the inference problem, we can still solve it efficiently in practice. *Interestingly, constraints do not add any overhead to our learning algorithm of* HMM$^{\text{CCM}}$.

The rest of this section is organized as follows: we first review HMM, a commonly used model for structured prediction. Then we show how to derive the supervised training algorithm for HMM$^{\text{CCM}}$. In the third part of this section, we describe CoDL, a semi-supervised learning algorithm for CCMs and apply it to HMM$^{\text{CCM}}$.

3.1 Hidden Markov models: a review

Hidden Markov Model (HMM) is one of the most commonly used models for sequence labeling. An HMM is a generative model parameterized by $P(y_i|y_{i-1})$ (the transition probabilities between consecutive hidden states), $P(x_i|y_i)$ (the emission probabilities of observing x_i from the state y_i) and $P(y_1)$ (the prior probabilities). In the discussion below, we denote the HMM parameters as Θ. HMM models the joint probability $P_\Theta(\mathbf{y}, \mathbf{x})$ of a series of tokens $\mathbf{x}$ of length T and a sequence assignment $\mathbf{y}$ as follows:

$$P_\Theta(\mathbf{y}, \mathbf{x}) = P(y_1) \prod_{i=2}^{T} P(y_i|y_{i-1}) \prod_{i=1}^{T} P(x_i|y_i). \tag{7}$$

Note that while the independence assumptions allow a compact representation of the joint probability and tractable inference algorithms, HMMs capture only the "local" behavior of a given task. For example, the transition table represents the probability of the assignment to y_i given the assignment to y_{i-1}. HMMs do not model "long distance" relationships such as the relationship between the first assignment y_1 and the last assignment y_T, nor they model global properties of the output sequence.

Standard training of an HMM is done by finding the parameters that maximize the likelihood of the labeled instances and can be efficiently done with partial counting over the training data (Rabiner and Juang 1986). That is, learning an HMM is equivalent to finding a Θ which maximizes the log likelihood $\sum_{j=1}^{l} \log P_\Theta(\mathbf{x}^j, \mathbf{y}^j)$, where l is the number of training samples.

When evaluating the model on a new instance, the Viterbi algorithm (Rabiner and Juang 1986) can be used to efficiently find the most likely assignment $\mathbf{y}$ defined as:

$$\arg\max_{\mathbf{y}} P_\Theta(\mathbf{y}|\mathbf{x}). \tag{8}$$

Past works have shown that the prediction problem in HMMs can be viewed as a linear model over "local" features (Roth 1999; Collins 2002). That is, one can show that

$$\arg\max_{\mathbf{y}} P_\Theta(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \log P_\Theta(\mathbf{x}, \mathbf{y}) = \arg\max_{\mathbf{y}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}), \tag{9}$$

where $\mathbf{w}$ is a weight vector and Φ represents the feature functions. Therefore, we can convert the probability tables Θ of an HMM into a linear function represented by $\mathbf{w}$ with appropriate feature functions. In this representation, the feature function $\Phi(\mathbf{x}, \mathbf{y})$ is expressed as a set of features, consisting of "prior features", $\Phi_p(y_1)$, "transition features", $\Phi_t(y_i, y_{i-1})$, and "emission features", $\Phi_e(x_i, y_i)$ (Roth 1999). In other words, there exists a one-to-one mapping between the active features and the associated probability representation, which can be rewritten in the form of a linear function.

$$\theta = \arg\max_{\theta} P(\mathcal{D}|\theta) = \arg\max_{\theta} \sum_{j=1}^{l} \log P\left(\mathbf{x}^j, \mathbf{y}^j|\theta\right),$$

where θ is the set of parameters that represent the prior, emission and transmission distributions.

3.2 HMM$^{\text{CCM}}$: supervised training

Assume that we have m constraints $C_1, C_2, \ldots, C_m$. In HMM$^{\text{CCM}}$, in order to combine statistical models and constraints (Eq. (3)), we adopt the idea of "product of experts" (Hinton 1999), where the HMM is the expert that predicts the probability of the label assignment, and the constraints component downgrades solutions that violate the constraints. This defines a new scoring function:

$$\begin{aligned}\Omega\left(\mathbf{x}^j, \mathbf{y}^j\right) &= \text{HMM Probability} \times \text{Constraint Violation Score} \\ &= P_\Theta\left(\mathbf{x}^j, \mathbf{y}^j\right) \prod_{k=1}^{m} \prod_{i=1}^{T_j} P(C_k = 1)^{c_{k,i}^j} P(C_k = 0)^{1-c_{k,i}^j}, \end{aligned} \tag{10}$$

where Θ are the parameters of the HMM, T_j represents the number of tokens in the sentence $\mathbf{x}^j$, $c_{k,i}^j$ is a binary variable equal to 1 if the label assignment to y_i^j violates the constraint C_k with respect to partial assignment $\mathbf{y}_{[1\ldots i-1]}^j$, and $C_k = 1$ indicates the event that the constraint C_k is violated. It is important to notice that the constraint violation score captures the "degree of violation" by counting the penalty multiple times.

The new scoring function $\Omega(\mathbf{x}^j, \mathbf{y}^j)$ augments the original HMM with the constraints we have. It is important to notice that Eq. (10) is a CCM. We can write $\log \Omega(\mathbf{x}^j, \mathbf{y}^j)$ in the form of (3) as follows:

$$\begin{aligned}\log \Omega\left(\mathbf{x}^j, \mathbf{y}^j\right) &\equiv \hat{f}_{w,\rho}\left(\mathbf{x}^j, \mathbf{y}^j\right) \\ &= \mathbf{w}^T \Phi\left(\mathbf{x}^j, \mathbf{y}^j\right) + \sum_{k=1}^{m} \log \frac{P(C_k = 1)}{P(C_k = 0)} \sum_{i}^{T_j} c_{k,i}^j + c \\ &= \mathbf{w}^T \Phi\left(\mathbf{x}^j, \mathbf{y}^j\right) - \sum_{k=1}^{m} \rho_k d_{C_k}\left(\mathbf{x}^j, \mathbf{y}^j\right) + c, \end{aligned} \tag{11}$$

where $\rho_k = -\log \frac{P(C_k=1)}{P(C_k=0)}$, $d_{C_k}(\mathbf{x}^j, \mathbf{y}^j) = \sum_i^{T_j} c_{k,i}^j$ and c is a constant which does not affect the inference results. Note that the definition of the terms $d_{C_k}(\mathbf{x}^j, \mathbf{y}^j)$ matches the one defined earlier in Eq. (2).

To train HMM$^{\text{CCM}}$ we need to find $\mathbf{w}$ and ρ that maximize the new scoring function

$$\sum_{j=1}^{l} \log \Omega\left(\mathbf{x}^j, \mathbf{y}^j\right) = \sum_{j=1}^{l} \hat{f}_{w,\rho}\left(\mathbf{x}^j, \mathbf{y}^j\right). \tag{12}$$

It is worth noting several things. First, despite the fact that we use probabilities extensively in the scoring function, the function in Eq. (12) itself does *not* represent the log likelihood of the dataset, since the augmented model does not have a likelihood interpretation. Nevertheless, it is still a smooth concave function and its optimal value can be determined by setting the gradient to zero. Algorithm 1 describes the training procedure in detail. Interestingly, the solution resembles the standard HMM model. In fact, we can estimate the prior probability, transition probability and emission probability in exactly the same way as in HMM. For the constraint violation part, a simple derivation shows that the optimal value for $P(C_k = 1)$ is

obtained by

$$P(C_k = 1) = \frac{\sum_{j=1}^{l} \sum_{i}^{T_j} c_{k,i}^{j}}{\sum_{j=1}^{l} T_j}. \tag{13}$$

Note that the training procedure is "inference-free" in the sense that it is only based on partial counting. We do not need to solve any inference problems during the training but apply the constraints only at the test phase. In Sect. 2.2 we discuss several alternatives for efficient approximate and exact solutions to the inference problem. This completes the machinery for supervised training and inference in HMM$^{\text{CCM}}$.

Algorithm 1 Supervised Learning HMM$^{\text{CCM}}$. The algorithm optimizes the objective function $\sum_{j=1}^{l} \log \hat{f}_{w,\rho}(\mathbf{x}^j, \mathbf{y}^j)$ defined in Eq. (12)

Require: $\mathbf{L}$: labeled training set, $\{C_k\}_{k=1}^{m}$: a set of constraints

Calculate Θ, the parameters of the HMM model with traditional HMM training.
Obtain $\mathbf{w}$ by applying the transformation on Θ described in Roth (1999), Collins (2002)
for $k = 1 \ldots m$ (constraint index) **do**
  for $j = 1 \ldots |\mathbf{L}|$ (training instance index) **do**
    for $i = 1 \ldots T_j$ (token position) **do**
      $c_{k,i}^{j} \leftarrow \hat{C}_k(\mathbf{x}^j; y_1^j, \ldots, y_i^j)$
    end for
  end for
end for
$P(C_k = 1) = \frac{\sum_{j=1}^{l} \sum_{i}^{T_j} c_{k,i}^{j}}{\sum_{j=1}^{l} T_j}$.
$\rho_k = -\log \frac{P(C_k=1)}{P(C_k=0)}$.
return $\mathbf{w}$, ρ

3.3 HMM$^{\text{CCM}}$: semi-supervised learning

Acquiring labeled data is a difficult and expensive task. Therefore, an increased attention has been recently given to semi-supervised learning, where large amounts of unlabeled data are used to improve models learned from a small training set (Yarowsky 1995; Blum and Mitchell 1998; Collins and Singer 1999; Thelen and Riloff 2002; Haghighi and Klein 2006).

Before we discuss unsupervised and semi-supervised training in HMM$^{\text{CCM}}$, it is useful to introduce some new notation. Throughout this section, we assume, for the sake of simplicity and wlog, that there is only one unlabeled observed input example, $\mathbf{x}_U$, with associated unobserved output sequence $\mathbf{h}$. When we use a model or an oracle to assign values to $\mathbf{h}$, we call the pair $(\mathbf{x}_U, \mathbf{h})$ pseudo-labeled data. Also, to avoid notation overload, we assume that we have one labeled and one unlabeled instance. This allows us to drop the sums of the form $\sum_j P(\mathbf{x}^j, \mathbf{y}^j)$ and write instead $P(\mathbf{x}, \mathbf{y})$. We note that this is done without loss of generality and for notational convenience only.

Traditionally, unsupervised and semi-supervised learning are done with the Expectation Maximization (EM) algorithm (Dempster et al. 1977; Borman 2004). Given only the unlabeled data $\mathbf{x}_U$, the EM algorithm is an iterative method for finding the model parameters θ

that maximize the objective function[3]

$$\theta^* = \arg\max_{\Theta} \log P_\Theta(\mathbf{x}_U) = \arg\max_{\Theta} \log \sum_h P_\Theta(\mathbf{x}_U|\mathbf{h}) P_\Theta(\mathbf{h}).$$

Unfortunately, while it is possible to estimate the full distribution $P(\mathbf{h}|\mathbf{x}_U)$ when the model only captures "local decisions", it is very difficult to estimate this distribution when long distance, expressive constraints are used.[4]

In order to alleviate the difficulty of estimating the full distribution in the presence of constraints, we maximize the function $\log \Omega(\mathbf{x}_U, \mathbf{h})$ over both **the model parameters** $(\mathbf{w}, \rho)$ and **the label assignment h**, which is equivalent to solving the problem:

$$\left(\mathbf{w}^*, \rho^*, \mathbf{h}^*\right) = \arg\max_{\mathbf{w},\rho,\mathbf{h}} \log \Omega(\mathbf{x}_U, \mathbf{h}) = \arg\max_{\mathbf{w},\rho,\mathbf{h}} \hat{f}_{w,\rho}(\mathbf{x}_U, \mathbf{h}).$$

In contrast to EM, which only maximizes the likelihood of the unlabeled data by marginalizing over hidden variables, we search for the best pseudo-label **h** and the model parameters $(\mathbf{w}, \rho)$ at the same time.

Our objective function can be optimized as follows (with initial **w** and ρ):

1. (Inference) Fix **w** and ρ, and optimize **h**.
   The solution for **h** with fixed **w** and ρ is given in Eq. (11) and can be found using the algorithms described in Sect. 2.2. In other words, **h** is the solution of the following optimization problem:

$$\mathbf{h} \leftarrow \arg\max_{\mathbf{h}} \hat{f}_{w,\rho}(\mathbf{x}_U, \mathbf{h}) = \arg\max_{\mathbf{h}} \mathbf{w}^T \Phi(\mathbf{x}_U, \mathbf{h}) - \sum_{k=1}^{m} \rho_k d_{C_k}(\mathbf{x}_U, \mathbf{h}).$$

2. (Learning) Fix **h**, and optimize **w** and ρ.
   The solution for optimizing **w** and ρ can be obtained by applying Algorithm 1 on the pseudo-labeled data $(\mathbf{x}_U, \mathbf{h})$.

By the definition in Eq. (11), both steps are guaranteed to increase the objective function. Again, note that this procedure has an advantage over EM: it does not need to compute the conditional probability distribution, but only to get the best assignment **h** for the example $\mathbf{x}_U$.

In HMM$^{\text{CCM}}$, the weight vector and the penalty vector resemble the probability distributions defined in Sect. 3.3 so they can be estimated easily. As in EM, the objective function is not convex. Therefore, it is essential to have a good starting point.

Since a good starting point is necessary, we move our focus to "semi-supervised learning" and use a small number of labeled examples to initialize the weight vector. One key difference between semi-supervised learning and unsupervised learning is that we need to balance the labeled training data and the unlabeled training data in order to have the best results. It is known that traditional semi-supervised training can degrade the learned model's performance (Nigam et al. 2000; Cozman et al. 2003). Nigam et al. (2000) has suggested balancing the contribution of labeled and unlabeled data to the parameters. In our algorithm, we use a similar intuition, but instead of weighting data instances, we introduce a smoothing

[3] Recall that θ can be rewritten in the form of CCMs using **w** and ρ.

[4] Ganchev et al. (2010) proposed to use expectation constraints to address this issue. See Sect. 7 for a discussion.

parameter γ which controls the convex combination of the *models* induced by the labeled and unlabeled data.

Algorithm 2 provides the pseudocode of the semi-supervised algorithm we called **CoDL** (COnstraint-Driven Learning) in Chang et al. (2007). We note that CoDL is a general procedure, and as such, can and will be applied to models other than HMM^{CCM}in later sections.

As is often the case in semi-supervised learning, the algorithm can be viewed as a process that improves the model by generating feedback through *labeling* unlabeled examples. Our algorithm pushes this intuition further, in that the use of constraints allows us to better exploit domain information as a way to label, along with the current learned model, unlabeled examples. Given a small amount of labeled data and a large unlabeled pool, our framework initializes the model with the labeled data and then repeatedly:

1. Uses *constraints* and the learned model to label the instances in the pool (line 5)
2. Updates the model using newly labeled data (line 8).

This way, we can generate *better* "training" examples during the semi-supervised learning process. Note that line 8 also performs the linear combinations among models with the parameter γ.

Algorithm 2 Constraint driven learning algorithm, which uses constraints to guide semi-supervised learning

Require: **L**: labeled training set, **U**: unlabeled dataset N: learning cycles
γ: balancing parameter with the supervised model,
$\{C\}$: a set of constraints,
learn(.): a supervised learning algorithm

1: Initialize $(\mathbf{w}, \rho) = (\mathbf{w}_0, \rho_0) = \mathbf{learn}(\mathbf{L})$.
2: **for** N iterations **do**
3: $\quad \mathbf{T} = \emptyset$
4: $\quad$ **for** $\mathbf{x} \in \mathbf{U}$ **do**
5: $\quad\quad \hat{\mathbf{h}} \leftarrow \arg\max_{\mathbf{y}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}) - \sum_{k=1}^{m} \rho_k d_{C_k}(\mathbf{x}, \mathbf{y})$
6: $\quad\quad \mathbf{T} = \mathbf{T} \cup \{(\mathbf{x}, \hat{\mathbf{h}})\}$
7: $\quad$ **end for**
8: $\quad (\mathbf{w}, \rho) = \gamma(\mathbf{w}_0, \rho_0) + (1 - \gamma)\mathbf{learn}(\mathbf{T})$
9: **end for**

CoDL uses constraints as prior knowledge in the semi-supervised setting. We later show that prior knowledge plays a crucial role when the amount of labeled data is limited. CoDL makes use of CCMs, which provide a good platform for combining the learned models with prior knowledge. It is very important to note that $CoDL$ can naturally be presented as a general purpose semi-supervised learning algorithm for any CCM model. For example, in Sect. 6 we show how to apply CoDL to averaged structured perceptron within the CCM framework.

It is interesting to note that in the absence of constraints, CoDL reduces to "hard-EM", which only finds the best assignment in every step. To further illustrate the difference between CoDL, "hard-EM" and (soft) EM, consider the problem of unsupervised part-of-speech tagging. In (soft) EM, we do not find the most likely label assignment given the data as part of the training procedure. On the other hand, when estimating the model parameters, we smoothed over all possible label assignments weighted by their likelihood. When

we run "hard-EM", we get the most likely label assignment as part of the procedure. Like "hard-EM", CoDL also finds only the best assignment during the learning processing. However, unlike "hard-EM", CoDL makes use of constraints to guide the learning process. More comparisons between CoDL, "hard-EM" and EM will be discussed in Sect. 5.3.

3.4 HMM^{CCM} versus $\text{HMM}^{\text{CCM}}_{\infty}$

We would like to stress again that HMM^{CCM} is just one algorithm of applying CCMs to Hidden Markov Models. One simple variation is to use "hard constraints" in CCM (denoted $\text{HMM}^{\text{CCM}}_{\infty}$, given that the penalty is infinity). The advantage of using hard constraints in CCMs is that we do not need to learn the penalty vector ρ, and the learning algorithm for the supervised setting is exactly the same as for HMM. The semi-supervised learning algorithm for HMM^{CCM}(Algorithm 2) can be directly applied to $\text{HMM}^{\text{CCM}}_{\infty}$. The disadvantage of $\text{HMM}^{\text{CCM}}_{\infty}$ is that it always enforces the constraints, which can in fact be violated in the gold data. See Sect. 5.5 for more comparisons between these two CCM approaches.

4 Tasks and data

In this section we introduce two information extraction problems which we used to evaluate the models and ideas presented in this paper. In both problems, given input text, a set of predefined fields is to be identified. Since the fields are typically related and interdependent, these kinds of applications provide a good test case for an approach like ours (the data for both problems is available at: http://cogcomp.cs.illinois.edu/page/resources/data[5]).

The first task is to identify fields from citations (McCallum et al. 2000). The data originally included 500 labeled references, and was later extended with 5,000 unannotated citations collected from papers found on the Internet (Grenager et al. 2005). Given a citation, the task is to extract the fields that appear in the given reference. There are 13 possible fields including author, title, location, etc.

To gain an insight into how the constraints can improve the model accuracy and guide semi-supervised learning, assume that the sentence shown in Fig. 1 appears in the unlabeled data pool. Part (a) of the figure shows the correct labeled assignment and part (b) shows the assignment labeled by an HMM trained on 30 labeled samples. However, if we apply the

(a) [*AUTHOR* Lars Ole Andersen .] [*TITLE* Program analysis and specialization for the C programming language .] [*TECH-REPORT* PhD thesis ,] [*INSTITUTION* DIKU , University of Copenhagen ,] [*DATE* May 1994 .]

(b) [*AUTHOR* Lars Ole Andersen . Program analysis and] [*TITLE* specialization for the] [*EDITOR* C] [*BOOKTITLE* Programming language] [*TECH-REPORT* . PhD thesis ,] [*INSTITUTION* DIKU , University of Copenhagen , May] [*DATE* 1994 .]

Fig. 1 Error analysis of an HMM model. The labels are underlined to the right of each open bracket. The correct assignment is shown in (**a**). The predicted assignment (**b**) violates some constraints, most obviously, the punctuation marks

[5]Note that we used different training-test split in our experiments than Grenager et al. (2005).

Table 1 The list of constraints used in the citations domain. Some constraints are relatively difficult to represents in traditional models

Citations	
Start	The citation can only start with author or editor.
AppearsOnce	Each field must be a consecutive list of words, and can appear at most once in a citation.
Punctuation	State transitions must occur on punctuation marks.
BookJournal	The words *proc, journal, proceedings, ACM* are *JOURNAL* or *BOOKTITLE*.
Date	Four digits starting with 20xx and 19xx are *DATE*.
Editors	The words *ed, editors* correspond to *EDITOR*.
Journal	The word *journal* are *JOURNAL*.
Note	The words *note, submitted, appear* are *NOTE*.
Pages	The words *pp., pages* correspond to *PAGE*.
TechReport	The words *tech, technical* are *TECH_ REPORT*.
Title	Quotations can appear only in titles.
Location	The words *CA, Australia, NY* are *LOCATION*.

constraint that state transition can occur only on punctuation marks, the same HMM will result in the correct labeling (a). Therefore, by adding the improved labeled assignment we can generate better training samples during semi-supervised learning. In fact, the requirements on punctuation marks are only some of the constraints that can be applied to this problem. The set of constraints we used in our experiments appears in Table 1. Note that some of the constraints are non-local and are very intuitive for people, yet it is very difficult to inject this knowledge into most models.

The second problem we consider is extracting fields from advertisements (Grenager et al. 2005). The dataset consists of 8,767 advertisements for apartment rentals in the San Francisco Bay Area downloaded in June 2004 from the Craigslist website. In the dataset, only 302 entries have been labeled with 12 fields, including *size, rent, neighborhood, features*, and so on. The data was preprocessed using regular expressions for phone numbers, email addresses and URLs. The list of the constraints for this domain is given in Table 2. We implement some global constraints and include unary constraints which were largely imported from the list of seed words used in Haghighi and Klein (2006). We slightly modified the seed words due to differences in pre-processing.

5 Experimental results

We empirically verify the effectiveness of combining constraints and statistical models in this section. The experiments are designed to answer the following series of research questions.

(1) **How important is it to add knowledge into statistical models?** More specifically:

- How does HMM^{CCM} perform compared to the original HMM?
- How efficient is it to use constraints as a supervision resource?

Note that these two questions address different aspects of using constraints in CCMs. The first question addresses the amount of improvement obtained by adding constraints. The second question, on the other hand, addresses the issue of using constraints as a *supervision resource* compared to labeling examples.

Table 2 The list of constraints used in the advertisements domain. Some constraints are relatively difficult to represents in traditional models. *Phone*, *Email* and *Money* are tokens corresponding to phone numbers, email addresses and monetary units, which were identified in text using regular expressions. This preprocessing was done before applying any training algorithms

Advertisements	
FieldLength	Each field must be at least 3 words long.
Punctuation	State transitions can occur only on punctuation marks or the newline symbol.
Address	The words *address, carlmont, st, cross* are *ADDRESS*.
Available	The words *immediately, begin, cheaper* are *AVAILABLE*.
Contact	The words **Phone*, *Email** are *CONTACT*.
Features	The words *laundry, kitchen, parking* are *FEATURES*.
Neighborhood	The words *close, near, shopping* are *NEIGHBORHOOD*.
Photos	The words *http, image, link* are *PHOTOS*.
Rent	The words *$, *Money** are *RENT*.
Restrictions	The words *smoking, dogs, cats* are *RESTRICTIONS*.
Roomates	The words *roommates, respectful, drama* are *ROOMMATES*.
Size	The words *sq, ft, bdrm* are *SIZE*.
Utilities	The words *utilities, pays, electricity* are *UTILITIES*.

(2) **How do our CCM training algorithms compare against other algorithms?**

- Is it beneficial to use the CoDL algorithm with the hard-EM approach, which finds the best assignment of the hidden variables, as opposed to EM, which calculates the full posterior distribution?
- How is the CCM approach compared to other approaches?

First, we ask the question about what are the benefits of using CoDL as opposed to the standard EM and hard EM algorithms. We then compare CoDL to other approaches of encoding long distance relationships. Note that we can often design a heavily engineered and tailored model for a specific task. However, this process is challenging and time consuming, and must be repeated for every new task. On the other hand, CCMs provide an easy to use model-specification language that works for all tasks. Therefore, we compare HMM^{CCM} to several tailored models to see if our general purpose model can match the performance of a specifically designed model. It is natural to expect that a tailored model will perform at least as well as CoDL; however, if CoDL matches the performance of a tailored model, we consider it a success. We also compared the results to recent approaches of using expectation constraints (Bellare et al. 2009).

(3) **What are the properties of CCMs and CoDL?** These questions include:

- In HMM^{CCM}, do we need to learn the penalty vector ρ?
- What is the utility of each constraint in our experiments?
- How important is it to tune γ in CoDL?

Among all of the research questions, the most important one is to verify whether adding constraints can improve the models or not. Again, while CCMs are not the only way to incorporate constraints, they provide a nice interface so that users do not need to invent a tailored model for every task.

The results reported in this and the following sections are token-level accuracies, which were averaged over 5 randomly generated training sets. We tested on a fixed test set and a

fixed development set, both containing 100 labeled samples. When semi-supervised learning algorithms are used, we use 1000 held-out unlabeled examples as part of our training data in both domains. This setting was first used by Grenager et al. (2005), Chang et al. (2007), Haghighi and Klein (2006) and then used by many other works. In the semi-supervised setting we ran 5 iterations of CoDL. The reason that we choose to run only 5 iterations is that our semi-supervised learning procedure usually converges very fast (see Sect. 5.3 for more details).

5.1 How does HMM^{CCM} perform compared to the original HMM?

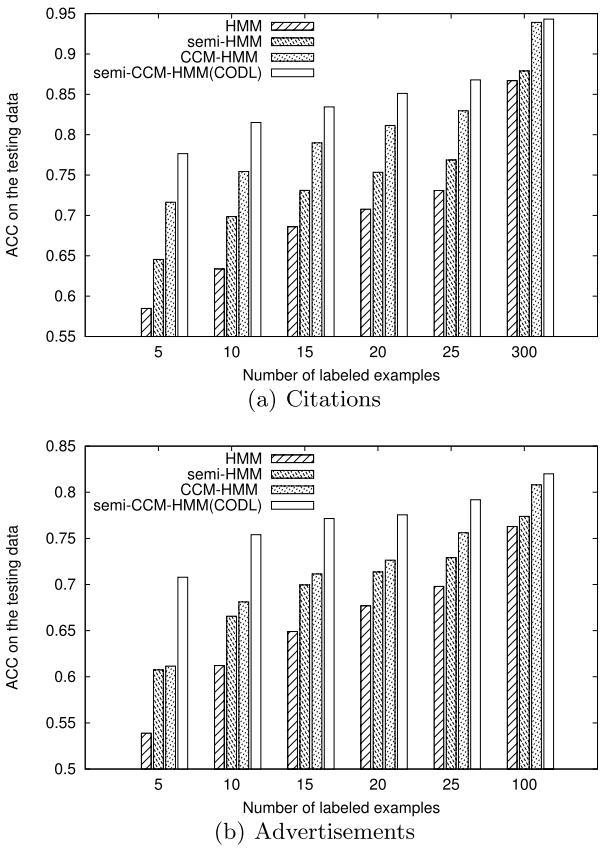
To see the impact of using constraints, we compare HMM and HMM^{CCM}in Table 3. The effect of applying constraints is significant: for example, when there are only 5 labeled examples, the constraints push the accuracy from 58 % to 71 % in citation domain and from 53 % to 61 % in advertisement domain. The results for more data points are shown in Fig. 2.

In the semi-supervised setting, adding constraints improves the HMM models more dramatically. One interesting result (see Table 3) is that with small amount of labeled data, the benefit of applying constraints is greater in the semi-supervised setting than that in the supervised setting. That is, with 5 labeled samples, in the advertisements domain, applying constraints in the supervised setting reduces the error rate by 15.47 % while applying constraints in the semi-supervised setting reduces the error rate by 25.58 %. Similarly, on the citations domain, applying the constraints reduces the error rate by 31.69 % in the supervised setting, while in the semi-supervised setting, the error rate decreases by 36.96 %. This result highlights the utility of using constraints in semi-supervised setting.

While with small amounts of labeled data, the majority of improvement comes from guiding semi-supervised learning with constraints, the situation is reversed when more labeled data is available. In this scenario, the parameters of the basic model are learned fairly well and semi-supervised learning cannot improve them further. In this case, most of the improvement comes from applying the constraints, while the utility of semi-supervised learning is limited. Nevertheless, for the advertisements domain, semi-supervised learning with constraints outperforms the supervised protocol with constraints by 1.2 % (82.00 versus 80.80) even when 100 labeled samples are available.

Table 3 The impact of using constraints for supervised and semi-supervised learning (generative HMM). Note that while semi-supervised HMMs performs much better than supervised HMMs, using constraints still improves the semi-supervised HMMs significantly. The numbers in the brackets denote error reduction over similar algorithm without constraints

# labeled samples	Supervised		Semi-supervised	
	HMM	HMM^{CCM}	HMM	HMM^{CCM}
Citations				
5	58.48	71.64 (31.69 %)	64.55	77.65 (36.96 %)
10	63.37	75.44 (32.94 %)	69.86	81.51 (38.67 %)
20	70.78	81.15 (35.49 %)	75.35	85.11 (39.61 %)
300	86.69	93.92 (54.29 %)	87.89	94.32 (53.07 %)
Advertisements				
5	53.90	61.16 (15.74 %)	60.75	70.79 (25.58 %)
10	61.21	68.12 (17.80 %)	66.56	75.40 (26.42 %)
20	67.69	72.64 (15.32 %)	71.36	77.56 (21.63 %)
100	76.29	80.80 (19.02 %)	77.38	82.00 (20.40 %)



(a) Citations

(b) Advertisements

Fig. 2 The utility of constraints in semi-supervised setting

5.2 How efficient is it to use constraints as a supervision resource?

We would like to view the results in the previous section from a different perspective: we can acquire knowledge either by adding constraints or adding more labeled samples. Here we view the "constraints" as a supervision resource rather than a part of the models and examine the utility of adding constraint as opposed to adding more labeled data.

The results in Table 3 clearly suggest that adding constraints is more efficient than adding labeled samples. Note that the model driven by constraints and 20 labeled samples outperforms the traditional HMM trained with 100 labeled samples on the advertisements domain and is only slightly worse compared to the traditional HMM trained with 300 labeled samples on the citations domain.

Figure 3 strengthens the claim of using constraints as a supervision resource. The left figure shows that in the citations domain, the semi-supervised HMM^{CCM}achieves, with 25 labeled samples, similar performance to the supervised version without constraints with 300 labeled samples. The right figure distinguishes the impact constraints made in the training phase and in the test phase. Semi-HMM(CoDL) represents the results where we train the HMM model with CoDL but do *not* apply the constraints in the test phase. The goal of this experiment is to see how well can CoDL guide the statistical component of the CCMs (in this case, the statistical component is HMM). Semi-HMM (CoDL) and HMM have the

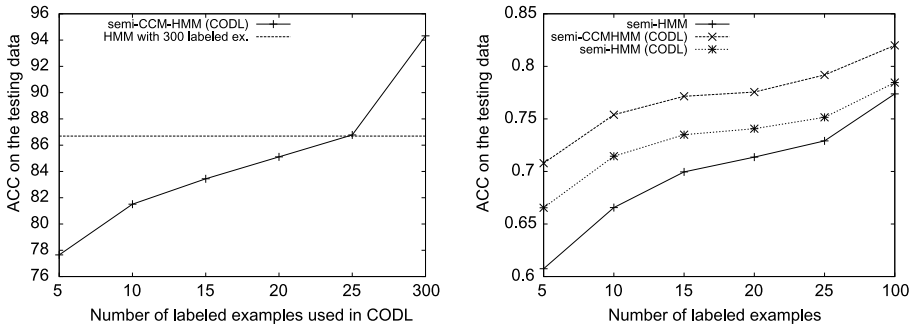


Fig. 3 Using constraints as supervision resource. *Left:* In the *citations* domain, with 25 labeled citations, our semi-supervised algorithm performs competitively to the supervised version trained on 300 samples. *Right:* The *ads* domain. Note that in semi-HMM (CoDL), we train the HMM model with CoDL, but *do not* apply the constraints in the test phase. The goal of this experiment is to see how well can CoDL guide the statistical component of the CCMs (in this case, the statistical component is HMM). The superior performance of semi-HMM (CoDL) shows that CoDL indeed can successfully guide the HMM

same expressivity, but the former is trained with constraints (using CoDL) while the latter is trained without using constraints. The superior performance of semi-HMM (CoDL) shows that CoDL can indeed guide the HMM successfully. This demonstrates the value of constraints as an additional supervision resource.

In other words, injecting constraints into the model requires design effort, but we believe that the increased expressivity of the model is well worth the effort. For example, applying constraints to the basic HMM trained on 300 labeled samples, improves the accuracy from 86.66 % to 94.03 %. We wanted to get a rough estimate on the number of additional labeled samples that are needed to achieve similar performance with the traditional HMM. Since the performance of the semi-supervised model on the citations domain is 94.51 %, we assume that the labels assigned to the unlabeled examples are fairly accurate. Therefore, we used our final model to label the unlabeled data and appended it to the training set. This way, we had 1300 labeled samples, which we used to train an HMM without constraints. The resulting accuracy was 88.2 %, still far from 94.51 %.

Moreover, when we trained the HMM on the training *and the test set* (400 labeled samples altogether), the resulting accuracy was 95.63 %. That is, even after seeing the test samples, the HMM does not have the expressivity to learn the true concept. On the other hand, when the constraints are applied, the accuracy goes up to 99.22 %. Therefore, we speculate that the basic HMM is simply unable to capture the expressive declarative aspects of the problem, no matter how much labeled data is available.

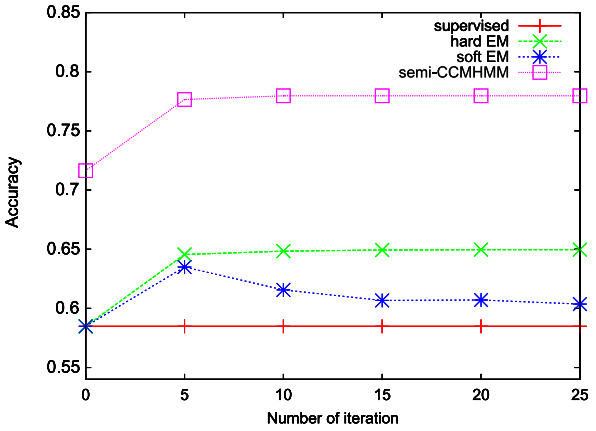
5.3 Is it beneficial to use the CoDL algorithm with the hard-EM approach, which finds the best assignment of the hidden variables, as opposed to EM, which calculates the full posterior distribution?

Expectation Maximization is the standard semi-supervised learning algorithm for generative models. In Sect. 3.3 we showed that our semi-supervised learning algorithm has an objective function which is different from that of EM, and very similar in spirit to hard-EM, which only find the best assignment instead of finding the full posterior distribution. In fact, when constraints are not used, our learning procedure is identical to hard-EM. The difference between EM and hard-EM is that the former requires to predict a full posterior distribution

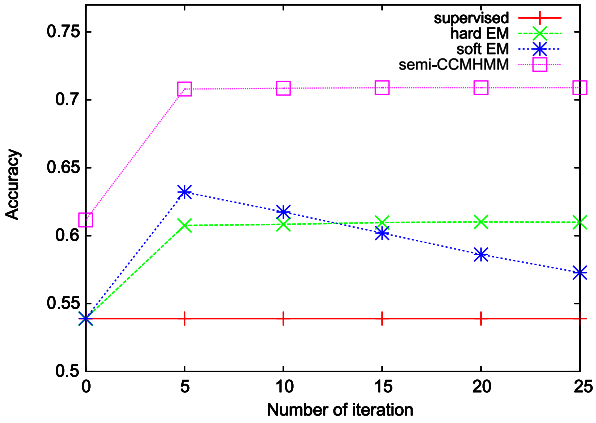
according to the parameters, while the latter one only requires finding the best assignment. It is important to note that when hard constraints are used, it is very difficult to calculate the distribution $P(\mathbf{y}|\mathbf{x})$ because of the long distance relationships. Note that one can relax constraints by transforming them into expectation constraints and make calculating posterior tractable (Ganchev et al. 2010). Please see the discussion in Sect. 7 for more details. In Sect. 5.4, we compare our algorithm to published results of a framework called alternating projections (AP), which uses expectation constraints in an EM-like algorithm.

In this section, we compare three approaches: **EM**, **hard-EM** and **semi-HMM$^{\mathbf{CCM}}$**. Note that the difference between hard-EM and semi-HMM$^{\mathrm{CCM}}$ is the use of the constraints. The experimental results of using 5 labeled examples are in Fig. 4. For all of the approaches, we put more weight on the labeled data and less weight on the unlabeled data ($\gamma = 0.9$). Putting more weights on the supervised model helps all three approaches.

First, in our experiments, we find that the accuracy of the EM approach degrades as the number of iterations grows in Fig. 4. While EM tries to maximize the log likelihood of the observed variables, it does not necessary mean that the model will get better performance as the number of the iterations grows (Liang and Klein 2008). This observation is consistent with Merialdo (1991), Liang and Klein (2008), Collins-Thompson (2009). Therefore, in our



(a) Citations, 5 training samples



(b) Ads, 5 training samples

Fig. 4 The test accuracy vs. number of iterations of semi-supervised learning in the citations and ads domains with 5 labeled examples. Note that the difference between hard-EM and semi-HMM$^{\mathrm{CCM}}$ is the usage of the constraints. See the text for more discussion

experiments, we find that while the EM approach can be better than the hard-EM approach (see Fig. 4(b), number of iterations equals to 5), the hard-EM approach is generally more stable as the number of iterations grows. In fact, we hardly see any change for the hard-EM approach after 5 iterations, and this is the reason why we choose to run only 5 iterations for the semi-supervised learning algorithms.

Recall that Algorithm 2 is similar to the hard-EM procedure but allows using constraints. Figure 4 shows that the **semi-HMM$^{\text{CCM}}$** approach is significantly better than both **EM** and **hard-EM**. Figure 4 also demonstrates that it is important to use constraints in the semi-supervised learning algorithms when the size of the labeled data is small.

5.4 How is the CCM approach compared to other approaches?

In this section, we compare the proposed approach to other existing approaches. First, we compare to a "tailored model" with a semi-CRF model (Sarawagi and Cohen 2004), which integrates the long distance relationships and local relationships together in one model. Second, we compared CCM to the alternating projection (AP) framework (Bellare et al. 2009), which can be considered as a discriminative special case of the Posterior Regularization framework (Ganchev et al. 2010). Note that the AP framework is not a "tailored model" given that the AP framework also keeps the baseline model and the constraints separately.

Comparison with CRF and semi-CRF We have seen that constraints allow us to capture properties of the problem which HMM cannot capture. However, it can be argued that we can modify an HMM model with certain amount of work. For example, if we segment the text on punctuation marks, and use a multinomial emission model for each state, we can capture the "transition on punctuation marks" constraint. The question is whether such a tailored model will perform significantly better than an off-the-shelf HMM with our constraints. Before we go into further discussion, we note that an HMM cannot be tailored to capture all the constraints; for example, the constraint "each field can appear only once" cannot be injected into an HMM model by tailoring segmentation, emission and transition components. Also, we argue that it is significantly more time consuming to engineer and implement a tailored model (particularly with semi-supervised training) than to take an off-the-shelf model and downweigh the output space with constraints violation penalties. Moreover, the tailored model we consider in this section can also be augmented with additional declarative constraints. In fact, the tailored model can be considered as an instance of CCMs, but with a tailored way to inject the constraints. Therefore, if the more general way of injecting constraints which we propose in this paper is competitive with the tailored model, we consider it a success for CCMs.

We choose Semi-Markov CRF (semi-CRF) (Sarawagi and Cohen 2004) as our tailored model competitor. Semi-CRF operates on a segment level rather than on a token level. That is, we define a segmentation to be $s = (s_1, \ldots, s_T)$ where each s_i $(1 \leq i \leq T)$ is a triple (t_i, u_i, l_i) with t_i denoting the segment beginning, u_i the segment end, and l_i—the assigned label. Training a semi-CRF involves finding the weights, and inference involves finding the segmentation which optimizes the following function:

$$P(s|\mathbf{x}, W) = \frac{1}{Z(\mathbf{x})} \exp^{\mathbf{w}^T \Phi(\mathbf{x}, s)}$$

where $\mathbf{x}$ is the input sequence, s is the segmentation, $\Phi(\mathbf{x}, s)$ are the features extracted from the segmentation s of $\mathbf{x}$, and $Z(\mathbf{x}) = \sum_{s'} \exp^{\mathbf{w}^T \Phi(\mathbf{x}, s')}$.

Table 4 Comparison between HMM^{CCM} and tailored models in the citations domain. Note that semi-CRF is a *supervised* learning algorithm and that semi-CRF^{+} uses additional features such as the segmentation length. Also note that in this table, both HMM^{CCM} and semi-CRF only use the "Punctuation" constraint and all the other models do not use any constraint. We only show the results for the citation domain, because we could not tune the semi-CRF model to perform competitively on the advertisements domain using the same features

Training instances	HMM	HMM^{CCM}	CRF	Semi-CRF	Semi-CRF^{+}
5	58.48	63.68	51.43	50.69	60.14
10	63.37	66.88	54.61	50.38	62.51
20	70.78	77.52	63.92	62.96	72.22
300	86.69	93.35	89.09	92.46	94.60

This allows the model to extract segment-level features, such as string edit distance to a multi-token dictionary of entities, and an average field length. Semi-CRFs exploit the fact that in many applications, adjacent tokens take the same label, an assumption that indeed holds in our data as well. For our problems, semi-CRFs have an attractive quality—they allow to inject segment-level features like "segment length" and "segment ends on a punctuation mark". Another attractive property of semi-CRFs is that the computational penalty paid for adding the segment-level expressivity when compared to first-order CRF is linear in L, the maximal segment length. We stress that there are important differences between semi-CRF (which tailors the training and inference to accommodate segment-level features), order-L CRF and CRF with a constrained output space. Comparing these models is outside the scope of this paper; the interested reader is referred to Sarawagi and Cohen (2004) and Roth and Yih (2005).

Semi-CRFs were originally proposed for the problem of named entity recognition (Cohen 2004; Sarawagi and Cohen 2004) with significant performance gains due to the ability of the model to capture inexact segment-level string matching to gazetteers. The computational penalty is high—the maximal length of a named entity was assumed to be 4, so the inference for semi-CRF is 4 times slower than for token-level first order CRF. In our problems, however, the maximum field length for citations was 100 tokens, and the maximum field length for the advertisements was 200 tokens, making the training and the inference of the model prohibitively slow.

Therefore, we compared the behavior of the competing models with a single constraint—"transition on punctuation marks". This constraint is readily injected into the semi-CRF by adding a feature indicating whether a segment ends with a punctuation mark. We compared the following models: HMM, HMM^{CCM}, CRF and semi-CRF. The HMM and the CRF models are without constraints. HMM^{CCM}and semi-CRF use a single constraint—"transition on punctuation marks". The default implementation of semi-CRF makes use of multiple additional features, including token normalization, token prefixes, suffixes, whether the token contains only digits, and also, most importantly—the segment length. To make a fair comparison, we removed most of these features, and used the same token-level features as in HMM. However, we were curious to see how much the segment length feature can improve the performance, particularly since it comes built in with the tailored model design. Therefore, we have 2 flavors of semi-CRFs: semi-CRF and semi-CRF^{+}, one with and one without the segment length feature.

The results are summarized in Table 4. We note that CRF is a discriminative model, therefore, as it is often the case, it performs worse than the generative model (HMM) when there is little training data and outperforms the HMM when a lot of training data is available

Table 5 Comparison to Alternating Projections (Bellare et al. 2009), a discriminative special case of Posterior Regularization (Ganchev et al. 2010). The AP results are taken from Bellare et al. (2009), while the CCM results are from Table 3

# labeled	AP-T	AP-I	Semi-HMM^{CCM}
5	75.6	74.6	77.65
20	85.4	85.1	85.11
300	94.0	94.3	94.32

(Ng and Jordan 2001). Furthermore, semi-supervised training in discriminative models is substantially harder. We also note that the injection of constraints in a generic way as done in the CCM framework improves the HMM performance from 86.66 to 93.35 with 300 labeled samples. Injecting the constraint "state transitions can occur only on punctuation marks" by tailoring CRF, improves the performance from 89.09 for CRF to 92.46 for semi-CRF, and including the additional feature of segment length in semi-CRF$^+$ further improves the performance to 94.60 on the citation domain with 300 labeled samples. Therefore, we see that although the tailored model has some potential, injecting constraints in the CCM framework actually brings bigger performance gains.

It is important to note that while semi-CRF performed very well on the citations domain, we failed to tune it to perform competitively on the advertisements domain. We suspect that the reason is the fact that we use very simple features in our CRF model, since the advertisements domain is a lot more difficult than the citation domains.

Comparison with expectation constraints approaches The Posterior Regularization Framework (PR) (Ganchev et al. 2010) is very related to the CCM-based CoDL algorithm (see Sect. 7 for a discussion). It is motivated by the observation that, while calculating posterior with hard constraints can be difficult, calculating posterior distribution with expectation constraints can be tractable, with a careful design.

The Alternating Projections framework (Bellare et al. 2009) can be considered as a special case of PR, tailored for discriminative models. In (Bellare et al. 2009), the authors perform experiments on the citation dataset in a very similar setting to the one we use in this paper, although their training-test data split is a bit different. Table 5 is created using our results in Table 3 and citing their reported results in (Bellare et al. 2009).

In Table 5, there are two AP approaches: AP-T uses the test dataset as the unlabeled dataset while AP-I uses another unlabeled dataset to bootstrap the results. Note that the performance of AP models are quite similar to those of the CCM models. This is so, despite the fact that the baseline model used by the AP models is a much stronger CRF model than both the CRF or and HMM baseline models we built in this paper. Their baseline CRF model is built with many different additional features including token features (identity, token prefixes, token suffixes and character n-grams), lexicon features (presence of a token in a lexicon of author names, journal names, etc.), regular expressions (common patterns for years and page numbers), and other bi-gram features.

5.5 In HMM^{CCM}, do we need to learn the penalty vector ρ?

Previous works (Punyakanok et al. 2005b; Roth and Yih 2005) have used "hard" constraints to disallow any label assignments that violate them. In the problems considered in this work, several gold assignments in the training set violate the constraints. Therefore, it seems beneficial to learn a constraint penalty vector ρ. As mentioned in Sect. 3.4, HMM^{CCM} is just

Table 6 Comparison of using hard and soft constraints in semi-supervised learning

Training samples	5	10	20	300
(a)-Citations				
semi-HMM^{CCM}	77.65	81.51	85.11	94.32
semi-HMM$^{CCM}_{\infty}$	78.18	81.11	85.16	92.80
(b)-Advertisement				
semi-HMM^{CCM}	70.79	75.40	77.56	82.00
semi-HMM$^{CCM}_{\infty}$	69.91	73.46	75.25	79.59

Table 7 Utility of hard constraints on the citations domain; supervised and semi-supervised setting with 5 training examples

Constraint	Supervised	Semi-supervised
None	**58.48**	**64.55**
Start	58.52	64.52
AppearsOnce	58.69	65.92
Punctuation	63.68	71.23
BookJournal	58.96	64.68
Date	61.50	66.76
Editor	58.70	64.77
Journal	58.66	64.73
Note	58.55	64.61
Pages	58.77	64.68
TechReport	58.73	64.43
Title	59.66	65.54
Location	58.81	64.97
ALL	**71.64**	**77.65**

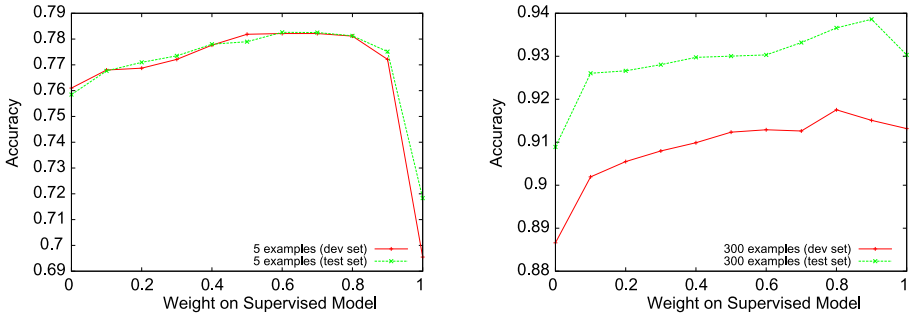
one instance of a CCM model, and we can also have a CCM version of HMM that makes use of hard constraints HMM$^{CCM}_{\infty}$. Table 6 shows that with sufficient amount of labeled data, HMM^{CCM} (learning with soft constraints) outperforms HMM$^{CCM}_{\infty}$ in both the citations and the advertisements domains.

5.6 What is the utility of each constraint in our experiments?

To highlight the impact of each constraint, in the following experiments, rather than learning the penalty of constraints violation from the data, we have enforced hard constraints. Tables 7 and 8 show the contribution of each constraint individually. Table 7 shows that the constraint *Start* (which requires the citations to start with either author or editor) actually hurts the performance in the semi-supervised setting. The constraint *AppearsOnce* hurts the performance in the supervised setting, but improves it significantly in the semi-supervised setting. Global constraints, such as *Punctuation*, improve the performance the most. Another interesting result is that while local constraints do not improve the performance significantly (even in the semi-supervised setting), when combined with the global constraints, they lead to significant performance improvements. While Tables 7 and 8 show the impact of using hard constraints, it is worthwhile to note that the soft constraints perform better (see Table 6).

Table 8 Utility of hard constraints on the advertisements domain; supervised and semi-supervised setting with 5 training examples

Constraint	Supervised	Semi-supervised
None	**53.90**	**60.75**
FieldLength	54.38	63.85
Address	53.95	60.85
Available	53.96	60.77
Contact	53.90	60.75
Features	54.20	60.84
Neighborhood	53.90	60.75
Photos	53.90	60.67
Rent	53.89	60.80
Restrictions	54.27	60.79
Roomates	53.90	60.78
Size	54.22	61.11
Punctuation	58.64	68.81
Utilities	54.05	60.89
All	**61.16**	**70.79**



(a) 5 examples (b) 300 examples

Fig. 5 The performance of the HMM semi-supervised algorithm with constraints on the citations domain. The x axis represents the weight γ of the supervised model. When the weight is 0, there is no smoothing at all and the model is equivalent to pure semi-supervised training. When weighting parameter is 1, the results will be equivalent to those of a purely supervised model

5.7 How important is it to tune γ in CoDL?

It is well known (for example, Cozman et al. 2003) that semi-supervised learning can degrade the performance when the assumptions of the model do not hold on the data. One way to overcome this problem is to reweigh labeled and unlabeled samples. Recall that in analogy to Nigam et al. (2000), when performing semi-supervised learning, we use the weighted average of the models trained on labeled and unlabeled data (see Sect. 3.3).

Figure 5 summarizes the effect of weighting parameter γ in line 8 of Algorithm 2. As expected, when the amount of the labeled data is increased, the model performs better with smaller values of γ. Note that in the experiments reported in this paper, we do not adjust the weighting parameter for different sizes of labeled data. We always use a fixed weighting

parameter $\gamma = 0.9$, which is not the optimal value for small training sets (for example, for 5 labeled examples).

6 CCM-infused Structured Perceptron (SP^{CCM})

Section 5 focused on using a maximum likelihood based training of HMM. In this section we show that the techniques discussed in Sect. 5 can generalize to other models by introducing a CCM-infused structured perceptron (SP^{CCM}) algorithm.

Recall that the objective function of a CCM (Eq. (3)) is:

$$f_{\Phi,C}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{x},\mathbf{y}) - \sum_{k=1}^{m} \rho_k d_{C_k}(\mathbf{x},\mathbf{y}). \tag{14}$$

Let us first ignore the constraints part and denote $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \mathbf{w}^T \Phi(\mathbf{x},\mathbf{y})$. In structured perceptron, the training of the weight vector $\mathbf{w}$ is done with the following update rule:

$$\mathbf{w}_{new} = \mathbf{w}_{old} + \Phi\left(\mathbf{x},\mathbf{y}^*\right) - \Phi(\mathbf{x},\hat{\mathbf{y}}), \tag{15}$$

where $\hat{\mathbf{y}}, \mathbf{y}^*$ are the predicted and correct values of $\mathbf{y}$, respectively.

The challenge of adapting CCMs to structured perceptron lies in training the complete model: the weight vector $\mathbf{w}$ and the violation penalty vector ρ. When the constraints are hard, ρ is fixed to infinity, and need not be tuned. However, even when $\rho = \infty$, two strategies can be used for training the weight vector $\mathbf{w}$. The difference is whether we want to consider the constraints when predicting the values $\hat{\mathbf{y}}$ during training. In one strategy, we use the same procedure during training and inference—this training scheme is referred to as *Inference Based Training* (IBT) in Punyakanok et al. (2005b). Surprisingly, it turns out that it is often better to ignore the constraints when predicting the values $\hat{\mathbf{y}}$ during training, and to enforce the constraints only during inference. This approach is called *Learning Plus Inference* (L+I) in Punyakanok et al. (2005b). Finally, when the constraints are soft, it is possible to treat the constraint violation penalties as features, and proceed with the traditional structure perceptron training framework. We call this update rule *Joint Inference Based Training* (JIBT). More formally, Algorithm 3 gives the pseudocode of the three training strategies for SP^{CCM}.

Table 9 compares these approaches and the baseline structured perceptron without constraints, denoted by L. We note that all the results were obtained with averaged perceptron, which performs better than perceptron without averaging. It can be seen that while IBT seems like a reasonable strategy, it does not perform well. L + I performs better than the baseline structured perceptron and IBT. Moreover, consistently with Punyakanok et al. (2005b), for a small number of examples, L + I outperforms all other algorithms, but when the amount of training data is large enough, learning the constraint violation penalties from the data (JIBT) achieves the best results.

Generally, semi-supervised training in discriminative models is challenging (Zhu 2006). Here, we propose an algorithm that has the CoDL flavor. It uses the following strategy: instead of averaging between the models of the supervised and the pseudo-labeled training data in line 8 of Algorithm 2, we first trained the perceptron on the pseudo-labeled data to obtain the weight vector $\mathbf{w}_U$, then initialized the weight vector to be $\mathbf{w}_0 \leftarrow (1-\gamma)\mathbf{w}_U$ (where γ is the smoothing parameter from Algorithm 2), and then continued to train w_0 on the labeled data. We refer to this algorithm as semi-SP^{CCM}. Table 10 summarizes the perfor-

Table 9 Comparison between discriminative learning strategies for average structured perceptron. Note that there are three strategies to learn a SP^{CCM}: L + I, IBT and JIBT. See the text for more details. Note that L + I outperforms *L* while *IBT* performs poorly. *JIBT* achieves the best results when enough data is used. Note that in the last row, we use 300 training examples for the citation domain and 100 examples for the advertisement domain

Tasks	Citations domain				Advertisements domain			
Algorithm	SP	SP^{CCM}			SP	SP^{CCM}		
Labeled samples	L	L + I	IBT	JIBT	L	L + I	IBT	JIBT
5	53.61	**68.66**	65.41	62.28	42.47	53.81	**54.87**	45.87
10	63.17	**75.57**	72.83	70.06	57.33	**64.24**	57.49	58.93
20	70.00	**79.72**	75.36	78.76	64.61	**68.01**	66.06	67.80
300 (100)	91.69	92.58	89.72	**94.57**	76.08	75.28	74.98	**79.36**

mance of these algorithms. Our heuristic algorithm improves the performance of structured perceptron for all cases except the citations domain with 300 labeled samples. A better and more principled semi-supervised learning algorithm for structured perceptron would be an interesting future research topic.

Algorithm 3 SP^{CCM} with three training strategies: L + I, IBT, and JIBT

Require: $\mathcal{D}$ is the training dataset, m is the number of constraints, M is the number of iterations, *rule* is the training strategy used for SP^{CCM}.

$$\forall k = 1 \ldots m : \rho_k = \begin{cases} \infty & \text{if (rule} = \text{L} + \text{I)} \vee \text{(rule} = \text{IBT)} \\ 0 & \text{if (rule} = \text{JIBT)} \end{cases}$$

for $t = 1 \ldots M$ **do**
  for $(\mathbf{x}, \mathbf{y}^*) \in D$ **do**

$$\hat{\mathbf{y}} = \begin{cases} \arg\max_{\mathbf{y}}[\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}) - \sum \rho_k d_{C_k}(\mathbf{x}, \mathbf{y})] & \text{if (rule} = \text{JIBT)} \vee \text{(rule} = \text{IBT)} \\ \arg\max_{\mathbf{y}}[\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y})] & \text{if (rule} = \text{L} + \text{I)} \end{cases}$$

    $\mathbf{w} = \mathbf{w} + \Phi(\mathbf{x}, \mathbf{y}^*) - \Phi(\mathbf{x}, \hat{\mathbf{y}})$
    if (*rule* = JIBT) then $\forall k : \rho_k = \rho_k + d_{C_k}(\mathbf{x}, \mathbf{y}^*) - d_{C_k}(\mathbf{x}, \hat{\mathbf{y}})$
  end for
end for

7 Related work

In this section we review selected publications related to the CCM framework. We first discuss several publications that can be considered as special cases of the CCMs framework. Other related publications are grouped into three different categories and the corresponding discussions are also included in this section.

Our work on CCMs builds on several works that can be considered as special cases of CCMs. In most cases, these works combine hard constraints with learning algorithms in the

Table 10 Comparison between HMM^{CCM} and SP^{CCM}. Our CoDL-like training algorithm improves the performance of structured perceptron (SP) for all cases except the citations domain with 300 labeled samples. Note that HMM^{CCM} is almost always superior in performance both when little labeled data and when all labeled data is available

Labeled samples	HMM^{CCM}	semi-HMM^{CCM}	SP	semi-SP^{CCM}
Citations domain				
5	71.64	77.65	62.28	65.84
10	75.44	81.51	70.06	73.22
20	81.15	85.11	78.76	79.20
300	93.92	94.36	94.57	93.75
Advertisements domain				
5	61.15	70.79	45.87	47.03
10	68.12	75.40	58.93	59.44
20	72.64	77.56	67.80	69.29
100	80.80	82.00	79.36	79.72

supervised setting. The first work in this line (Roth and Yih 2004) (extended in Roth and Yih 2007) suggests a formalism that combines constraints with linear models on information extraction tasks. They use linear inequalities and suggest Integer Linear Programming as the inference framework. Following Roth and Yih (2004, 2007), a series of works proposed and studied models that incorporate learned models with declarative constraints with successful applications in Natural Language Processing and Information Extraction, including semantic role labeling (Roth and Yih 2005; Punyakanok et al. 2005a, 2008), summarization (Clarke and Lapata 2006; Barzilay and Lapata 2006), generation (Marciniak and Strube 2005) and co-reference resolution (Denis and Baldridge 2007).

Most of these works use only hard constraints with the factored approach and with supervised classifiers. In contrast, we introduce soft constraints (modeled as the degree of violating a constraint), into the model and integrate constraints into semi-supervised learning, extending (Chang et al. 2007). In addition, we investigate different training paradigms for CCMs, both for the probabilistic component, and for the constraints component and provide a more rigorous analysis of using constraints in structured prediction tasks. The Never-Ending-Language-Learner (NELL) project provides a web scale experiment (Carlson et al. 2010) of the model proposed here for using constraints in semi-supervised learning algorithms. They highlight the importance of decoupling constraints from the model by showing that the constraints can have significant impact on the performance in the semi-supervised setting.

7.1 Capturing long distance relationships

In order to make the inference procedure of finding the best assignment tractable, most structured output prediction models only capture local relationships. While CCMs are designed to address this issue, several other approaches (most of which only focus on the supervised learning algorithms) have been proposed to address long distance relationships. For example, Collins (2000), Charniak and Johnson (2005), Toutanova et al. (2005) propose to use a two stage approach to address this issue: in the first stage, a local model is used to produce the k-best solutions and, in the second stage, a global model which captures long distance relationships is used to rerank the k-best solutions generated in the first stage. Since the global model only focuses on k assignments, modeling long distance relationships becomes tractable. However, this approach suffers from the problem of error propagation. If the k-best solutions produced by the local model does not contain the correct parse tree, it is impossible for the global model to find the correct solution.

Recently, Daumé and Marcu (2005), Kazama and Torisawa (2007), Huang (2008) proposed to use approximate inference procedure and let the weights of the long distance features guide the search procedure. This approach is similar to the beamsearch procedure we proposed in Sect. 2.2.3, in the sense that the search procedure is guided by the constraint penalties. Importantly, CCMs focus on injecting high level knowledge in the form of "first-order" like declarative features. For example, in this paper, we show that we can improve HMM very significantly with only 10 additional constraints. In contrast, lots of grounded features are used in Daumé and Marcu (2005), Kazama and Torisawa (2007), Huang (2008) to capture long distance relationships.

The named entity recognition system (Finkel et al. 2005) captures long distance relationships by using Gibbs sampling as their inference algorithm. The CCM framework is more general because (Finkel et al. 2005) only focuses on a specific type of long distance relationship while CCMs allow the use of general long distance relationships. It would be interesting to explore the possibilities of using sampling based methods such as Gibbs sampling methods in the CCM framework.

7.2 Expectation-maximization framework and posterior regularization

Posterior Regularization (PR) (Ganchev et al. 2010) is probably the work that is most related to the CoDL algorithm. It develops a CoDL-like approach, but is different in that it extends the EM algorithm by incorporating expectation constraints. While modeling the exact posterior distribution with hard constraints is expensive in general, PR relaxes the constraints to expectation constraints.

Let θ be the model parameters and $P_\theta(\mathbf{y}|\mathbf{x})$ be the conditional probability distribution according the θ. The E-step of the standard EM algorithm finds a posterior distribution q' according to:

$$q' = \arg\min_{q} D\big(q \parallel P_\theta(\mathbf{y}|\mathbf{x})\big),$$

where $D(q \parallel P_\theta(\mathbf{y}|\mathbf{x}))$ is the Kullback-Leibler divergence between the distributions q and $P_\theta(\mathbf{y}|\mathbf{x})$.

Given m constraints, Ganchev et al. (2010) assumes that the k-th constraint can be written in the following form:

$$f_k(\mathbf{x}, \mathbf{y}) \leq b_k.$$

The expectation constraints can be expressed by

$$E_q\big[\mathbf{f}(\mathbf{x}, \mathbf{y})\big] \leq \mathbf{b},$$

where $\mathbf{f}(\mathbf{x}, \mathbf{y})^T = \big[f_1(\mathbf{x}, \mathbf{y}) \ldots f_m(\mathbf{x}, \mathbf{y})\big]^T$, and $\mathbf{b} = \big[b_1 \ldots b_m\big]^T$.

The E-step in the PR framework then finds the best posterior probability which satisfies the expectation constraints

$$q' = \arg\min_{q: E_q[\mathbf{f}(\mathbf{x},\mathbf{y})] \leq \mathbf{b}} D\big(q \parallel P_\theta(\mathbf{y}|\mathbf{x})\big).$$

The CoDL algorithm proposed in Sect. 3.3 also uses an EM-like procedure. In fact, the main differences between CoDL and PR are in the E-step. There are two major differences: (1) The CoDL framework allows the use of hard constraints, while PR uses ex-

pectation constraints.[6] (2) While the PR algorithm obtains a distribution by minimizing the KL-divergence, the CoDL algorithm only finds the best assignment.

There exist other approaches which aim to combine constraints with statistical models. For example, Mann and McCallum (2008), Bellare et al. (2009) proposes a *Generalized Expectation Criteria*, which uses a different distance function in the E-step. Liang et al. (2009) proposes *Learning from Measurements*, which incorporates prior information about model posteriors from a Bayesian point of view. The PR paper Ganchev et al. (2010) explains the relationships between these frameworks very clearly. An empirical comparison between these frameworks is an interesting future research direction.

7.3 Injecting knowledge into graphic models

The combination of constraints and probabilistic graphical models has also been studied before from the probabilistic modeling perspective. For example, Dechter and Mateescu (2004) propose a combination of the Bayesian network model with a collection of deterministic constraints and call the resulting model a *mixed network*. They conclude that the deterministic constraints of a mixed network are handled more efficiently when maintained separately from the Bayes network and processed with special purpose algorithms. In addition, they find that the semantics of a mixed network are easier to work with and understand than an equivalent, "pure" Bayes network with deterministic constraints modeled probabilistically.

Similar in spirit, CCMs differ from the mixed networks by allowing the probabilistic portion of the model to represent an arbitrary *conditional* distribution, instead of a joint distribution (in the form of a Bayes network). We consider this an advantage, since CCMs do not waste their power to model the probability of input variables. Moreover, the current work proposes also algorithms for *learning* mixed representations in the form of CCMs.

Markov Logic Networks (MLN) (Richardson and Domingos 2006) is a probabilistic logic framework which uses logic to provide a convenient way of specifying a Markov Random Field, following a long tradition of works in this direction (Friedman et al. 1999; Ngo and Haddawy 1995; Kersting and Raedt 2000; Jaeger 1997).

MLN and CCMs are similar in that they both combine declarative knowledge into statistical models. The crucial difference between CCM and MLN is on the issue of model decomposition. MLN includes the expressive features (constraints) as part of the probabilistic model, while we propose factoring the model into a simpler probabilistic model with additional constraints, also expressed declaratively using first order logic like expressions (Rizzolo and Roth 2007). This has significant implications on the learning procedure. While in MLNs the learning problem is that of learning the whole joint model, in CCMs the goal is to learn a simpler model (e.g., an HMM in our experiments here) but nevertheless it supports doing inference with a more expressive model.

8 Conclusions

This paper provides a unified view of a framework aimed to facilitate decision making with respect to multiple interdependent variables the values of which are determined by learned

[6]The approximation we used in Sect. 2 is similar to the expectation constraint when the approximation collects statistics over the whole dataset. However, unlike the PR, we do not calculate the full posterior distribution.

statistical models. We proposed Constrained Conditional Models (CCMs), a framework that augments linear models with expressive declarative constraints as a way to support decisions in an expressive output space while maintaining modularity and tractability of training. Importantly, this framework provides a principled way to incorporate expressive background knowledge into the decision process. It also provides a way to combine conditional models, learned independently in different situations, along with declarative information to support coherent global decisions.

Acknowledgements This work was supported by NSF grant NSF SoD-HCER-0613885, DARPA funding under the Bootstrap Learning Program and by MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

References

Barzilay, R., & Lapata, M. (2006). Aggregation via set partitioning for natural language generation. In *Proceedings of HLT/NAACL*, June 2006.

Bellare, K., Druck, G., & McCallum, A. (2009). Alternating projections for learning with expectation constraints. In *Proceedings of uncertainty in artificial intelligence (UAI)*.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the annual ACM workshop on computational learning theory (COLT)* (pp. 92–100).

Borman, S. (2004). *The expectation maximization algorithm—a short tutorial*. Introduces the Expectation Maximization (EM) algorithm and fleshes out the basic mathematical results, including a proof of convergence. The Generalized EM algorithm is also introduced, July 2004.

Carlson, A., Betteridge, J., Wang, R. C., Hruschka, E. R. Jr., & Mitchell, T. M. (2010). Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on web search and data mining*.

Chang, M., Ratinov, L., & Roth, D. (2007). Guiding semi-supervision with constraint-driven learning. In *Proceedings of the annual meeting of the association for computational linguistics (ACL)*, Prague, Czech Republic, June 2007 (pp. 280–287). New York: Association for Computational Linguistics.

Charniak, E., & Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the annual meeting of the association for computational linguistics (ACL)*, Ann Arbor, Michigan (pp. 173–180). New York: ACL.

Clarke, J., & Lapata, M. (2006). Constraint-based sentence compression: an integer programming approach. In *Proceedings of the annual meeting of the association for computational linguistics (ACL)*, Sydney, Australia, July 2006 (pp. 144–151). New York: ACL.

Cohen, W. (2004). Exploiting dictionaries in named entity extraction: combining semi-Markov extraction processes and data integration methods. In *Proceedings of international conference on knowledge discovery and data mining (KDD)* (pp. 89–98).

Collins, M. (2000). Discriminative reranking for natural language parsing. In *Proceedings of the 17th international conference on machine learning* (pp. 175–182). San Francisco: Morgan Kaufmann.

Collins, M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the conference on empirical methods for natural language processing (EMNLP)*.

Collins, M., & Singer, Y. (1999). Unsupervised models for named entity classification. In *Proceedings of the conference on empirical methods for natural language processing (EMNLP)*.

Collins-Thompson, K. (2009). Reducing the risk of query expansion via robust constrained optimization. In *Proceedings of ACM conference on information and knowledge management (CIKM)* (pp. 837–846).

Cozman, F. G., Cohen, I., & Cirelo, M. C. (2003). Semi-supervised learning of mixture models. In *Proceedings of the international conference on machine learning (ICML)* (pp. 99–106).

Daumé, H., & Marcu, D. (2005). Learning as search optimization: approximate large margin methods for structured prediction. In *Proceedings of the international conference on machine learning (ICML)*, Bonn, Germany, 2005.

Dechter, R., & Mateescu, R. (2004). Mixtures of deterministic-probabilistic networks and their AND/OR search space. In *Proceedings of AUAI*, Arlington, VA, USA, 2004 (pp. 120–129). New York: AUAI Press.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, *39*, 1–38.

Denis, P., & Baldridge, J. (2007). Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of the annual meeting of the North American association of computational linguistics (NAACL)*.

Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the annual meeting of the association for computational linguistics (ACL)*, Morristown, NJ, USA, 2005 (pp. 363–370). New York: Association for Computational Linguistics.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)* (pp. 1300–1309).

Ganchev, K., Graça, J., Gillenwater, J., & Taskar, B. (2010). Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*.

Graca, J. V., Ganchev, K., & Taskar, B. (2007). Expectation maximization and posterior constraints. In *NIPS* (Vol. 20).

Grenager, T., Klein, D., & Manning, C. (2005). Unsupervised learning of field segmentation models for information extraction. In *Proceedings of the annual meeting of the association for computational linguistics*. New York: ACL.

Haghighi, A., & Klein, D. (2006). Prototype-driven learning for sequence models. In *Proceedings of HTL-NAACL*.

Hinton, G. (1999). Products of experts. In *Proceedings of the 9th international conference on artificial neural networks (ICANN99)* (pp. 1–6).

Huang, L. (2008). Forest reranking: discriminative parsing with non-local features. In *Proceedings of the annual meeting of the association for computational linguistics (ACL)*.

Jaeger, M. (1997). Relational Bayesian networks. In M. Kaufmann (Ed.), *Proceedings of the 13th conference on uncertainty in artificial intelligence* (pp. 266–273).

Kazama, J., & Torisawa, K. (2007). A new perceptron algorithm for sequence labeling with non-local features. In *Proceedings of the 2007 joint conference of EMNLP-CoNLL* (pp. 315–324).

Kersting, K., & Raedt, L. D. (2000). Bayesian logic programs. In J. Cussens & A. Frisch (Eds.), *Proceedings of the work-in-progress track at the 10th international conference on inductive logic programming* (pp. 138–155).

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the international conference on machine learning (ICML)*.

Liang, P., Jordan, M. I., & Klein, D. (2009). Learning from measurements in exponential families. In *Proceedings of the international conference on machine learning (ICML)*.

Liang, P., & Klein, D. (2008). Analyzing the errors of unsupervised learning. In *Proceedings of the annual meeting of the association for computational linguistics (ACL)*.

Mann, G., & McCallum, A. (2008). Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of the annual meeting of the association for computational linguistics (ACL)* (pp. 870–878).

Marciniak, T., & Strube, M. (2005). Beyond the pipeline: discrete optimization in NLP. In *Proceedings of the annual conference on computational natural language learning (CoNLL)*, Ann Arbor, MI, June 2005 (pp. 136–143). New York: Association for Computational Linguistics.

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the international conference on machine learning (ICML)*.

Merialdo, B. (1991). Tagging text with a probabilistic model. In *Proceedings of the international conference on acoustics, speech, and signal processing*.

Ng, A. Y., & Jordan, M. I. (2001). On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. In *The conference on advances in neural information processing systems (NIPS)* (pp. 841–848).

Ngo, L., & Haddawy, P. (1995). Probabilistic logic programming and Bayesian networks. In *Asian computing science conference* (pp. 286–300).

Nigam, K., Mccallum, A., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*(2/3), 103–134.

Punyakanok, V., Roth, D., & Yih, W. (2005a). The necessity of syntactic parsing for semantic role labeling. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)* (pp. 1117–1123).

Punyakanok, V., Roth, D., Yih, W., & Zimak, D. (2005b). Learning and inference over constrained output. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)* (pp. 1124–1129).

Punyakanok, V., Roth, D., & Yih, W. (2008). The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, *34*(2), 257–287.

Rabiner, L. R., & Juang, B. H. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine*, *3*(1), 4–16.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning Journal*, *62*(1–2), 107–136.

Rizzolo, N., & Roth, D. (2007). Modeling Discriminative Global Inference. In *Proceedings of the first international conference on semantic computing (ICSC)*, Irvine, CA, September 2007 (pp. 597–604). New York: IEEE.

Roth, D. (1999). Learning in natural language. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)* (pp. 898–904).

Roth, D., & Yih, W. (2004). A linear programming formulation for global inference in natural language tasks. In H. T. Ng & E. Riloff (Eds.), *Proceedings of the annual conference on computational natural language learning (CoNLL)* (pp. 1–8). New York: Association for Computational Linguistics.

Roth, D., & Yih, W. (2005). Integer linear programming inference for conditional random fields. In *Proceedings of the international conference on machine learning (ICML)* (pp. 737–744).

Roth, D., & Yih, W. (2007). Global inference for entity and relation identification via a linear programming formulation. In L. Getoor & B. Taskar (Eds.), *Introduction to statistical relational learning*. Cambridge: MIT Press.

Sarawagi, S., & Cohen, W. (2004). Semi-Markov conditional random fields for information extraction. In *The conference on advances in neural information processing systems (NIPS)* (pp. 1185–1192).

Thelen, M., & Riloff, E. (2002). A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the conference on empirical methods for natural language processing (EMNLP)*.

Toutanova, K., Haghighi, A., & Manning, C. D. (2005). Joint learning improves semantic role labeling. In *Proceedings of ACL 2005*.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervied methods. In *Proceedings of the annual meeting of the association for computational linguistics (ACL)*.

Zhu, X. (2006). *Semi-supervised learning literature survey*.