Constraints as Prior Knowledge

Ming-Wei Chang MCHANG21@UIUC.EDU
Lev Ratinov RATINOV2@UIUC.EDU
Dan Roth DANR@UIUC.EDU
Computer Science Department, University of Illinois at Urbana-Champaign

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 lead to expensive training and inference.

This paper presents Constrained Conditional Models (CCMs), a framework that augments probabilistic models with declarative constraints as a way to support decisions in an expressive output space while maintaining modularity and tractability of training. We develop, analyze and compare novel algorithms for training and inference with CCMs. Our main experimental study exhibits the advantage our framework provides when declarative constraints are used in the context of supervised and semi-supervised training of a probabilistic model.

1. Introduction

Decision making in domains such as natural language 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 (whose values are sought) and information sources, often referred to as input or observed variables.

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.

There exist several fundamentally different approaches to learning models that can assign values simultaneously to several interdependent variables (Punyakanok et al., 2005). Two extremes are to (1) completely ignore the output structure at the learning stage (by learning multiple independent models), while enforcing coherent assignments at the inference stage and (2) model, directly or indirectly, the dependencies among the output variables in the learning process and thus induce models that optimize a global performance measure. In the latter scenario, to allow efficient training and inference, assumptions on the probability distribution are made so that it is possible to factor the model into functions of subsets of the variables, yielding models such as Conditional Random Fields (CRFs) and Hidden Markov Models (HMMs).

However, 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 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. While efficient algorithms for leveraging a particular type of constraint can be developed, modeling of declarative non-local constraints this way is clearly very expensive. Moreover, a lot of parameters are being wasted in order to to learn something the model designer already knows.

This paper presents Constrained Conditional Models (CCMs). Generalizing and formalizing an approach introduced in (Roth & Yih, 2004; Roth & Yih, 2007), CCM is a framework that augments linear objective functions with declarative constraints as a way to support decisions in an expressive output space. CCMs inject the constraints *directly* instead of doing it indirectly via a probability model. CCM allows the use of expressive constraints while keeping models simple and easy to understand. Factoring the models by separating declarative constraints naturally brings up interesting questions and calls for novel training and inference algorithms, as we discuss in this paper.

One interesting perspective is that the declarative constraints can be viewed as domain-specific knowledge which can be injected into the model in the supervised and, more interestingly, in the semi-supervised setting. We develop a formalism for constraints-based learning within the CCM framework. Our protocol can be used in the presence of any learning model, including those that acquire additional statistical constraints from observed data while learning. We experiment and report results with two models: maximum likelihood HMM (Rabiner & Juang, 1986) and its discriminative counterpart–the structured perceptron (Collins, 2002). We exhibit significant reduction in the number of training examples required in two information extraction problems. The results show that our approach yields very good results even in the presence of a small number of labeled examples.

2. Linear Models for Sequence Labeling Tasks

Although the discussion in this paper can be applied to other types of problems, we mainly focus on an important type of structured prediction problems: sequence labeling tasks. Given $\mathbf{x}$ as a series of tokens, we denote x_i as the i-th token of $\mathbf{x}$. Assuming there are T tokens in $\mathbf{x}$, the assignment $\mathbf{y}$ can be written as $y_1, y_2, \dots y_T$, where y_i is the label for token x_i. The task in sequence labeling is to learn a model that can be used to predict the correct $\mathbf{y}$ given a new instance $\mathbf{x}$.

Linear models are the dominant family in machine learning, and 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: $f(\mathbf{x}, \mathbf{y}) = \sum w_i \phi_i(\mathbf{x}, \mathbf{y})$. Many different discriminative and generative learning algorithms can be represented as linear models. For example, models trained by Perceptron, naïve Bayes, and SVM, CRF and HMMs are linear models (Roth, 1999; Collins, 2002; Lafferty et al., 2001). Hidden Markov Model (HMM) is one of the most commonly used models for sequence labeling. 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:

$$\operatorname*{argmax}_{\mathbf{y}} \log P(\mathbf{y}|\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $\mathbf{w}$ is a weight vector and Φ represents the feature functions, is an equivalent representation of HMM.

3. Training and Inference with Constraints

Although, in general, the feature functions $\Phi(\mathbf{x}, \mathbf{y})$ used in Eq. 1 can represent any function of $\mathbf{x}$ and $\mathbf{y}$, it is typical to encode local relationships only, (as in the linear representation of HMMs (Roth, 1999; Collins, 2002; Lafferty et al., 2001)) for tractable inference. However, such restriction usually renders the feature functions not expressive enough to capture non-local dependencies present in the problem.

In this paper, we propose the **Constrained Conditional Model (CCM)**, which provides a *direct* way to inject prior knowledge into a conditional model, in the form of **constraints**. The idea is that combining simple models with *expressive* constraints is a more effective approach to making probabilistic models expressive. Note that we do not increase the feature space explicitly by adding more conjunctive features but rather directly incorporate the constraints by augmenting the simple linear models. Since within CCMs we combine declarative constraints, possibly written as first order logic expressions (Rizzolo & Roth, 2007), with learned probabilistic models, we can treat CCMs as a way to combine or bridge logical expressions and learning statistical models.

Note that by modeling the constraints directly, the inference problem, 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; it is possible to use exact methods such as integer linear programming or approximate inference methods that we found to give good results.

3.1. Model

The formal definition of CCM is as follows.

We assume (1) a set of feature functions $\Phi = \{\phi_i(\cdot)\}$, $\phi_i : \mathcal{X} \times \mathcal{Y} \to R$, which typically encode *local* properties of a pair (x, y). (And often, the image of ϕ_i is $\{0, 1\}$); (2) a small set of constraints $C = \{C_i(\cdot)\}$, $C_i : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ that encode predicates over a pair (x, y); (3) a set of functions $d_{C_i} : \mathcal{X} \times \mathcal{Y} \to R$ that measure the degree to which the constraint C_i is violated in (x, y).

A **Constrained Conditional Model** can be represented using two weight vectors, $\mathbf{w}$ and ρ. The score of an assignment $\mathbf{y} \in \mathcal{Y}$ on an instance $x \in \mathcal{X}$ is obtained by:

$$f_{\Phi,C}(\mathbf{x},\mathbf{y}) = \sum w_i \phi_i(\mathbf{x},\mathbf{y}) - \sum \rho_i d_{C_i}(\mathbf{x},\mathbf{y}). \quad (2)$$

A CCM then selects as its prediction:

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}} f_{\Phi,C}(\mathbf{x},\mathbf{y}). \quad (3)$$

Note that a CCM is not restricted to be trained with any particular learning algorithm. Similar to other linear models, specialized algorithms may need to be developed to train CCMs. Unlike standard linear models, we assume the availability of some prior knowledge, encoded in the form of *constraints*, when learning a CCM. When there is no prior knowledge, there is no difference between CCMs and linear models.

Although the two terms of Eq. 2 may appear similar, they are very different in several aspects. Essentially, a predicate $C(\mathbf{x},\mathbf{y})$ is viewed as a"first order logical expression", which is very different from features $\Phi(\mathbf{x},\mathbf{y})$. Due to their first order logic nature, the set of constraints is compact. (In our experiments, we only have about 10 constraints, compared to thousands of features in a feature vector). Moreover, $C(\mathbf{x},\mathbf{y})$ usually encodes long distance relationships among $\mathbf{y}$ variables, which cannot be captured by the feature functions $\Phi(\mathbf{x},\mathbf{y})$. For example, $C(\mathbf{x},\mathbf{y})$ might be "1, if all y_is in the sequence y are assigned different values, 0 otherwise", which is difficult to model using features.

Importantly, we separate the constraints from features in Eq. 2 because we know that the constraints should be trusted most of the time. Therefore, the penalties ρ can be fixed or handled **separately**. If we are confident about our knowledge, rather than learning the $\{\rho_i\}$, we can directly set them to ∞, thus enforcing the chosen assignment $\mathbf{y}$ to satisfy the constraints. It is important to note that although ρ_i is fixed, it may still impact the learning of the weights w_i (this point will be explained in detail in Section 3.3).

3.2. Inference with Constraints

In the earlier related works that made use of constraints, the constraints were assumed to be binary 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 a integer linear programming formulation(ILP) (Roth & Yih, 2004; Roth & Yih, 2007; Barzilay & Lapata, 2006; Clarke & Lapata, 2006). Intractable in principle, ILP proved to be quite successful in practice, since the constraints were very sparse (a small number of $\mathbf{y}$ variables present in each constraint) (Roth & Yih, 2007).

However, in our CCM formalism, rather than using binary constraints, we introduce a "degree of violation" to each constraint. The significance of this is that it is possible that a label assignment violates the constraints in more than one place. Therefore, if binary function is used, once the value is set to 1, the algorithm loses the ability to discriminate constraint violations in other locations of the same instance. Note that even with such a choice, ILP can still be applied to solve the inference problem Eq. 3. However, here we choose not to do it, but rather to approximate the degree of violation incrementally, by estimating it over an incomplete label assignment. This allows us to design a search procedure which finds an approximate solution to Eq. 3 efficiently. In this work, we rewrite the constraint function as:

$$d_{C_i}(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{T} \hat{C}_i(\mathbf{x}; y_1, \ldots, y_t),$$

where T is number of tokens in this instance, and $\hat{C}_i(\mathbf{x}; y_1, \ldots, y_t)$ is a binary function which approximates the predicate C_i, by computing it over the t-prefix of the assignment $\mathbf{y}$, $(\mathbf{x}; y_1, \ldots, y_{t-1})$.

We use this estimation to guide the search procedure for optimizing the objective function in Eq. 3 with partially labeled sequence. In this paper, we use beam search as our search procedure. A* search can be also applied here with admissible heuristic if the ρ_is are positive for all constraints. Note that this approximation methods may not work for all types of constraints. For example, constraints such as "label A must appear at least once in the sequence", do not have "degree" of violation. For these constraints, the function d_C is the identity function, essentially making them binary constraints; these constraints are examined only at the end of the search procedure.

3.3. Training with CCM

In this section, we propose and describe several approaches of training CCMs. There are two independent decisions to be made, leading to four different training strategies.

The first decision is whether we want to use *factored* approaches or *joint* approaches. *Factored* approaches treat the first term (feature term) and the second term (constraints term) of Eq. 2 separately. That is, $\mathbf{w}$ and ρ are learned *independently*. This approach is also referred to *Learning Plus Inference* (L+I) (Punyakanok et al., 2005), since we learn the models separately but put the constraints back into consideration at testing time. The *joint* approach, which we call *Inference Based Training* (IBT) (Punyakanok et al., 2005), learns $\mathbf{w}$ and ρ together during training by using the true objective function with both terms in Eq. 3.

The second decision is whether we want to use hard constraints or weighted constraints. Using hard constraints is equivalent to setting ρ to ∞; in this case, the notion of "degree" no longer exists, the constraints essentially become Boolean functions, and we do not output assignments which violate them. Using weighted constraints is important if we know that the prior knowledge does not hold all the time and it also means that we need to figure out the penalty ρ for each constraint from labeled data.

Training CCMs with factored approaches is simple, since factored approaches learn $\mathbf{w}$ and ρ independently. $\mathbf{w}$ can be learned with standard algorithms for training linear models. If we chose to use hard constraints, the training procedure is complete, given that the penalty of each constraint is infinity. In this paper, this approach is called **L+CI** (Learning Plus Constrained Inference) . However, it is often the case that the prior knowledge is not perfect, or that the weights for every constraint should be different. To figure out the penalty for each constraint, in this case, we count how many times it is violated in the labeled data, and reduce the penalty coefficients for those violated constraints (refer to (Chang et al., 2008) for details). This approach is called **L+wCI** (Learning Plus weighted Constrained Inference).

Alternatively, we can enforce the constraints during training as well as testing. In this approach, *Inference Based Training* (IBT), the constraints may be hard or soft, resulting in **CIBT** and in **wCIBT** respectively. Our IBT training algorithms are based on the Perceptron update rule.

The pseudocode for **CIBT** and **wCIBT** is given in Algorithm 1, which is similar to the perceptron algorithm. However, the constraints are taken into account during the training procedure. CIBT is a more conservative update rule than L+CI, since when the constraints term "corrects" the label assignment, no update will be performed. Note that when weighted constraints are used, the algorithm also updates the penalty ρ during the training procedure.

Since the constraints in Eq. 2 have non-local nature, we give up exact inference (with dynamic programming) and use beam search to find an approximate solution. The idea of using non-local features in perceptron was also explored in (Collins & Roark, 2004) that used

Algorithm 1 IBT training: CIBT & wCIBT

```
Require: $D$ is the training dataset, $K$ is the number
    of constraints, $M$ is the number of iterations
 1: for $i = 1 \ldots K$ do
 2:   $if(hardConstraints)$ then $\rho_i = \infty$ else $\rho_i = 0$
 3: end for
 4: for $i = 1 \ldots M$ do
 5:   for $(\mathbf{x}, \mathbf{y}^*) \in D$ do
 6:     $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}}[\sum w_i \phi_i(\mathbf{x}, \mathbf{y}) - \sum \rho_i d_{C_i}(\mathbf{x}, \mathbf{y})]$
 7:     $\mathbf{w} = \mathbf{w} + \Phi(\mathbf{x}, \mathbf{y}^*) - \Phi(\mathbf{x}, \hat{\mathbf{y}})$
 8:     if weightedConstraints then
 9:       $\rho = \rho + d_C(\mathbf{x}, \mathbf{y}^*) - d_C(\mathbf{x}, \hat{\mathbf{y}})$
10:     end if
11:   end for
12: end for
```

beam search for inference with application to syntactic parsing. Later, (H. Daumé & Marcu, 2005) extended this idea to other applications. While wCIBT uses a similar algorithm to assign weights to the constraints, it differs from (Collins & Roark, 2004; H. Daumé & Marcu, 2005) in the nature of the "features": there, a large number of weights are assigned to "propositional" non-local features in perceptron, while we assign a small number of weights to constraints that are high level, 'first order logic' predicates.

3.4. Semi-Supervised Learning with CCM

Acquiring labeled data is a difficult and expensive task. Therefore, an increasing 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 (Haghighi & Klein, 2006; Thelen & Riloff, 2002). In this section, we present COnstraint-Driven Learning (**CODL**), an algorithm that uses constraints as prior knowledge in semi-supervised setting (Chang et al., 2007) and show that prior knowledge plays a crucial role when the amount of labeled data is limited. CODL makes use of CCM, which provides a good platform to combine the learned models and prior knowledge.

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. *CODL* pushes this intuition further, in that the use of constraints allows us to better exploit domain information as a way to label unlabeled examples, along with the current learned model. Given a small amount of labeled data and a large unlabeled pool, *CODL* initializes the model with the labeled data and then repeatedly: **(1)** uses the learned model and the *constraints* to label the unlabeled instances, and **(2)** updates the model via the newly labeled data.

Algorithm 2 **CO**nstraint **D**riven **L**earning (CODL): Using constraints to guide semi-supervised learning.

```
Require: labeled training set L; unlabeled dataset U; N learning cycles; a balancing parameter with the supervised model $\gamma$; a set of constraints $\{C\}$; a supervised learning algorithm $learn(.)$
1: Init: $(\mathbf{w}, \rho) = (\mathbf{w}_0, \rho_0) = \mathbf{learn}_{[(\mathbf{w})\mathbf{CIBT}/\mathbf{L}+(\mathbf{w})\mathbf{CI}]}(\mathbf{L})$.
2: for $N$ iterations do
3:     $\mathbf{T} = \emptyset$
4:     for $\mathbf{x} \in \mathbf{U}$ do
5:         $(\mathbf{x}, \hat{\mathbf{y}}) = \mathbf{InferenceWithConstraints}(\mathbf{x}, \mathbf{w}, \rho, \{C_i\})$
6:         $\mathbf{T} = \mathbf{T} \cup \{(\mathbf{x}, \hat{\mathbf{y}})\}$
7:     end for
8:     $(\mathbf{w}, \rho) = (1 - \gamma)\mathbf{learn}_{[(\mathbf{w})\mathbf{CIBT}/\mathbf{L}+(\mathbf{w})\mathbf{CI}]}(\mathbf{T}) + \gamma(\mathbf{w}_0, \rho_0)$
9: end for
```

CODL is summarized in Algorithm 2. *CODL* initializes the model with traditional supervised learning on a small labeled set **L** (line 1). The supervised learning algorithm $learn_{[(w)CIBT/L+(w)CI]}(.)$ used in lines 1 and 8, learns $(\mathbf{w}, \rho)$ jointly if the wCIBT approach is used. If the L+wCI approach is used, it learns $\mathbf{w}$ independently from estimating ρ. If CIBT or L+CI is used, the learning algorithm $learn_{[(w)CIBT/L+(w)CI]}(.)$ always sets ρ to infinity.

Line 8 in the algorithm should be further clarified. (Nigam et al., 2000) shows that semi-supervised training can degrade the learned model's performance and suggests to balance the contribution of labeled and unlabeled data. The parameter re-estimation in line 8 uses a similar intuition, but instead of weighting data instances, we introduce a smoothing parameter γ which controls the convex combination of models induced by the labeled and unlabeled data. Unlike the technique mentioned above, which focuses on naïve Bayes, our method allows us to weight linear models generated by different learning algorithms. Due to space limitations we do not address several other important issues related to the algorithm, for more details, please refer to (Chang et al., 2008).

4. Experiments and Results.

We applied our approach to two information extraction tasks: extracting fields from citations and advertisements. Since in both problems, 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://L2R.cs.uiuc.edu/~cogcomp/Data/IE.tgz.`). Due to space restrictions, we omit the details of the datasets, and report only the main results, omitting the analysis constraints' utility, sensitivity to constraint violation penalty, etc. The reader is referred to (Chang et al., 2008) for additional details.

Citations	
Start	The citation must 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.*
...	...
TechReport	The words *tech, technical* are *TECH_REPORT.*
Title	Quotations can appear only in titles.
Location	The words *CA, Australia, NY* are *LOCATION.*

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

Table 1 illustrates the list of constraints for the citations domain. We measured token-level accuracy of the learned models and evaluated the impact of the constraints in the supervised and semi-supervised settings. Table 2 shows the results for HMM (trained in a maximum-likelihood way). The results highlight the effect of applying the constraints. A semi-supervised model driven by constraints and 20 labeled samples, using L+wCI, is competitive with the traditional HMM trained with 300 labeled samples.

Table 3 compares the discriminative approaches for structured perceptron (the baseline, without constraints, is denoted **L**). It can be seen that while CIBT seems like a reasonable strategy, it does not perform well. L+CI performs better than the baseline structured perceptron and CIBT. Moreover, consistently with (Punyakanok et al., 2005), for a small number of examples, L+CI outperforms all other algorithms while, when the amount of training data is large enough, learning the constraint violation penalties from the data (wCIBT) achieves the best results.

As observed already in the literature (see for example (Ng & Jordan, 2001)), with small amounts of labeled data, maximum-likelihood (ML) training approaches outperform discriminative ones. However, for sufficient amounts of data, and without constraints, the discriminative approach outperforms the ML approach. With 300 training samples on the citations domain, the structured perceptron achieves accuracy of 89.83% on the citations domain versus 86.35%, achieved by ML HMM when trained on the same amount of labeled data. However, when learning constraint violation penalties, the ML approach consistently outperformed the discriminative approach. One reason for that is that in L+wCI in ML approach, we assume that the constraints hold by default, and reduce the constraint violation penalty only when the labeled data violates the constraints. On the other hand, in the wCIBT approach in discriminative setting, we learn constraint violation penalties from scratch. More data must be needed for successful training. Moreover, despite trying several learning strategies, we could not achieve improvements with the semi-supervised training for the discriminative approach.

Citations(Maximum Likelihood HMM)				
	Supervised		Semi-Supervised	
#Train	HMM	L+wCI	HMM	L+wCI
5	58.48	70.85	64.39	**77.09**
10	68.61	75.11	70.34	**81.25**
20	70.81	81.31	75.83	**85.00**
300	86.66	94.08	87.80	**94.51**

Table 2. The impact of using constraints for supervised and semi-supervised learning (generative HMM) with 5,10,20,300 labeled training samples.

5. 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 probabilistic models. We proposed CCM, 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.

	Supervised setting. Structured Perceptron-Citations Domain			
#Train	L	L+CI	CIBT	wCIBT
5	50.14	**66.36**	64.79	61.65
10	59.90	**72.91**	68.52	69.64
20	68.26	77.28	72.79	**78.46**
300	89.83	91.63	87.83	**93.89**

Table 3. Comparison between discriminative learning strategies. L+CI outperforms *L* while *CIBT* performs poorly. *wCIBT* achieves the best results when enough data is used.

References

Barzilay, R., & Lapata, M. (2006). Aggregation via Set Partitioning for Natural Language Generation. *Proc.of HLT/NAACL.*

Chang, M., Ratinov, L., & Roth, D. (2007). Guiding semi-supervision with constraint-driven learning. *Proc. of the ACL.*

Chang, M., Ratinov, L., & Roth, D. (2008). Structured learning with constrained conditional models. *In Submission.*

Clarke, J., & Lapata, M. (2006). Models for sentence compression: A comparison across domains, training requirements and evaluation measures. *Proc. of ACL.*

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

Collins, M., & Roark, B. (2004). Incremental parsing with the perceptron algorithm. *Proc. of the ACL.*

H. Daumé, I., & Marcu, D. (2005). Learning as search optimization: approximate large margin methods for structured prediction. *Proc. of ICML.*

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

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

Ng, A. Y., & Jordan, M. I. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naïve bayes. *Proc. of NIPS* (pp. 841–848).

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning Journal*, *39.*

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

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

Rizzolo, N., & Roth, D. (2007). Modeling Discriminative Global Inference. *Proc. of the ICSC* (pp. 597–604). IEEE.

Roth, D. (1999). Learning in natural language. *Proc.of IJCAI.*

Roth, D., & Yih, W. (2004). A linear programming formulation for global inference in natural language tasks. *Proc. of CoNLL.*

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

Thelen, M., & Riloff, E. (2002). A bootstrapping method for learning semantic lexicons using extraction pattern contexts. *Proc. of EMNLP.*