AAAI '08

Learning and Inference with Constraints

Ming-Wei Chang, Lev Ratinov, Nicholas Rizzolo and **Dan Roth**
Computer Science Department
University of Illinois at Urbana-Champaign
{mchang21,ratinov2,rizzolo,danr}@uiuc.edu

Abstract

Probabilistic modeling has been a dominant approach in Machine Learning research. As the field evolves, the problems of interest become increasingly challenging and complex. Making complex decisions in real world problems often involves assigning values to sets of interdependent variables where the expressive dependency structure can influence, or even dictate, what assignments are possible. However, incorporating nonlocal dependencies in a probabilistic model can lead to intractable training and inference. This paper presents Constraints 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 further show that declarative constraints can be used to take advantage of unlabeled data when training the probabilistic model.

Introduction

Decision making in domains such as natural language processing is characterized by ambiguity and partial or imperfect information sources. Most significantly, complex decisions often involve assigning values to *sets* of interdependent variables where the expressive dependency structure can influence, or even dictate, what assignments are possible. To cope with these difficulties, one models the problem as a stochastic process involving both output variables (those whose values are sought) and the information sources, often referred to as input or observed variables.

There exist several fundamentally different solutions to learning models that can assign values simultaneously to interdependent variables, ranging from completely ignoring the output structure and learning local models, to decoupling the model learning stage from the task of respecting interdependencies among the output variables (sometimes, a necessity, if not all the components of a global decision can be trained together) to incorporating dependencies among the output variables into the learning process to directly induce models that optimize a global performance measure. 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).

However, in many problems dependencies among output variables have nonlocal 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 make 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, probabilistic modeling of declarative non-local constraints is clearly impractical.

This paper addresses the aforementioned problems by separating the probabilistic and the declarative aspects of the model. The proposed framework thus yields models that can support making decisions in expressive output spaces while maintaining the computational benefits of training simple models and keeping the models' representation 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 later in this paper.

Following (Roth & Yih 2004; 2007), in a series of works (Chang, Ratinov, & Roth 2007; Roth & Yih 2005; Punyakanok, Roth, & Yih 2005) we have studied models that incorporate learned models with declarative constraints. We applied our model to a variety of important Natural Language Processing problems such as Semantic Role Labeling (Koomen *et al.* 2005; Punyakanok, Roth, & Yih 2005; Roth & Yih 2005) and Information Extraction (Roth & Yih 2004; Chang, Ratinov, & Roth 2007), investigated tradeoffs between training paradigms (Punyakanok *et al.* 2005) and proposed a modeling language that facilitates programming systems that use conditional models with declarative constraints (Rizzolo & Roth 2007). This paper briefly introduces the model, summarizes the training and inference method and focuses on the use of constraints to guide supervised and semi-supervised models.

Training and Inference with Constraints

The dominant family of models used in machine learning are *linear models*, which can be presented as a weight vector w, corresponding to a set of feature functions $\{\phi\}$. For an input instance x and an output assignment y, the "score" of the instance can be expressed as a weighted sum of feature functions: $f(x,y) = \sum w_i \phi_i(x,y)$. During testing, given an unlabeled instance x, the aim is to *infer* the best assignment to the output variables, $y^* = \mathrm{argmax}_y \sum w_i \phi_i(x,y)$. Many different discriminative and generative learning algorithms learn linear models. For example, models trained by Perceptron, naïve Bayes, and SVM are linear models (Roth 1999). We can further define y over *structured labels* to make linear models well-suited for the structured prediction problems. Therefore, models produced by CRFs and HMMs also belong to linear models under the extended definition (Roth 1999; Collins 2002).

Although the linear models are popular, several restrictions exists. First, since the problem of finding the best assignment is not trivial, typically, the feature functions $\phi_i(x,y)$ are "local" and restricted to allow inference tractability (Collins 2002). However, such restriction usually renders the feature functions not expressive enough to capture the non-local dependencies present in the problem. While it is often possible to increase the expressivity of input-only features, including expressive dependencies among output values renders the model intractable. For instance, a second-order CRF will include many more features than a chain-style CRF, and is difficult to work with in realistic size domain, despite the fact that only a few of the new features are relevant.

The second shortcoming of linear models is that they lack a way to inject prior knowledge *directly*. The only way to inject prior knowledge is indirectly; by adding more features (Roth & Yih 2005). However, often a more direct infusion is preferable. For example, if we know of certain restrictions on the output space of y, it is better to implement those restrictions directly instead of hoping they will emerge through the learning of a model with extended features. This is especially important when there are not enough labeled examples for the models to learn the restriction properly.

We are not the first to address these issues. For example, (Dechter & 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.

To address these issues, we propose a superset of the linear model which we call the **Constrained Conditional Model (CCM)**. Similar to linear models, specialized algorithms may need to be developed to train CCMs. Unlike learning linear models, we assume that some prior knowledge exists in the form of *constraints* when learning a CCM. When there is no prior knowledge, there is no difference between CCMs and linear models.

Definition (CCM) *A CCM can be represented by two weight vectors, w and ρ, given a set of feature functions $\{\phi_i(\cdot)\}$ and a small set of constraints $\{C_i(\cdot)\}$. The score for an assignment $y \in \mathcal{Y}$ on an instance $x \in \mathcal{X}$ can then be obtained by*

$$f_C(x,y) = \sum w_i \phi_i(x,y) - \sum \rho_i C_i(x,y), \qquad (1)$$

where each $C_i : \mathcal{X} \times \mathcal{Y} \rightarrow \{0,1\}$ is a Boolean function indicating whether the joint assignment (x,y) violates i-th constraint. A CCM then selects $y^* = \mathrm{argmax}_y f_C(x,y)$ as its prediction.

CCM differentiates itself from the mixed network 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 as an advantage, since CCM does not waste its power to model the probability of input variables.

Note that a CCM is not restricted to any particular learning algorithm. In previous work, we have developed a modeling language for models of this type, providing flexible access to a wide variety of learning algorithms as well as the convenience of a first-order-logic-like syntax for declaratively expressing constraints. The constraints are automatically translated to linear inequalities for integer linear programming inference; see (Rizzolo & Roth 2007) for details.

Although the form of (1) is similar to linear models, the essence of our approach is very different. Specifically, rather than learning the $\{\rho_i\}$, we use external knowledge to set them[1]. Usually ρ is set to ∞, since we are confident about the knowledge. We can also set ρ to some positive value when the knowledge is not perfect. It is important to note that although ρ_i is fixed, it may still impact the learning of the weights w_i, since the assignment y selected by a CCM is that which scores highest according to (1) [2].

There are several advantages of using CCM instead of linear models. First of all, CCM provides a platform for encoding prior knowledge. As we will show later, this is especially important when there are not many labeled instances. Second, constraints can be much more expressive than the features used by linear models. Third, CCM can *simplify* the modeling of a complex structured output problem. Instead of building a model from complex features, CCM provides a way to combine "simple" learned models with a small set of "expressive" constraints. Importantly, combining simple models with constraints often results in better performance. For example, our top-ranking system in the CoNLL 2005 shared task uses a CCM approach and outperforms many systems built from complex features (Punyakanok, Roth, & Yih 2005).

[1]Previous works have shown that in many cases such external knowledge is available (Barzilay & Lapata 2006; Roth & Yih 2005; Chang, Ratinov, & Roth 2007).

[2]This point will be explained in details when we discuss **Training with CCM**.

Testing with CCM In general, the best assignment of (1) can be found via Integer Linear Programming (ILP). Although ILP is intractable in the limit, it can be quite successful in practice when applied to structured prediction problems, which are often sparse (Roth & Yih 2007). In fact, our ILP-based Semantic Role Labeler discussed in the next section is able to process several sentences per second with our unoptimized code. Beam search can be also used to find approximated solutions.

Other successful applications of our framework, with ILP as an inference algorithm, can be found in the literature as well. For example, (Barzilay & Lapata 2006) describes an automatic semantic aggregator that uses constraints to control the number of aggregated sentences and their lengths. (Marciniak & Strube 2005) describes a general ILP framework for solving multiple NLP problems simultaneously.

Training with CCM In this paper, we summarize two simple approaches to train a CCM. Other training algorithms are left for future research.

The first approach is to simply use standard, linear model training algorithms. That is, we discard the constraints at training time but enforce them at testing time. For instance, in our case study (Roth & Yih 2005), we show that training with CRF and then applying constraints at testing time is better than training and testing with a pure CRF model. This approach is referred to *Learning Plus Inference* (L+I) (Punyakanok *et al.* 2005). It is important to note that the left component of Eq. 1 need not be trained as a single linear model. It can be further decomposed to smaller models that are trained separately and put together with the constraints to form the objective function of Eq. 1 only at test time. This approach was used, for example in (Punyakanok, Roth, & Yih 2005), where each local model was a Perceptron. The question of how to decompose the objective function to simpler models and achieve both training efficiency and accurate test time decision is an important future question.

Alternatively, we can enforce constraints during training as well as testing, in an approach we call *Inference Based Training* (IBT). Our IBT training algorithms are based on the Perceptron linear model. When training with constraints, the weights w_i are updated only if $\mathrm{argmax}_y\, f_C(x, y)$ is incorrect. If the Perceptron model makes a mistake but the constraints term in (1) "corrects" the output, the training algorithm will not update w_i.

In principle, it may seem that IBT is preferable, since it takes into account constraints at training time and thus learn to optimize the desired objective function. However, (Punyakanok *et al.* 2005) shows that the IBT approach usually requires more labeled examples to outperform L+I, since the models are more complex. Therefore, when the local components of the left side of Eq. 1 are easy to learn as stand along models, L+I often performs better than IBT. Additionally, this happens also in the presence of limited supervision, which is often the case in natural language processing tasks.

Case Study: Semantic Role Labeling

Semantic Role Labeling (SRL) is the task of discovering the verb-argument structure for a given input sentence, that is, for each relation identifying the roles such as Agent, Patient or Instrument. The PropBank dataset (Palmer, Gildea, & Kingsbury 2005) provides labeled data with six main types of arguments labeled as A0-A5 and AA, which have different semantics for each verb (as specified in the PropBank Frame files) and a number of other types of modifiers and adjuncts. An example of an SRL-labeled sentence is:

```
[A0 I] [Verb left] [A1 my pearls] [A2
to my daughter-in-law].
```

Here A0 represents the leaver, A1 represents the thing left and A2 is the benefactor. In this case study, we only consider the core arguments, namely A0, A1, . . ., A5. All other chunks were labeled O.

We have developed two SRL systems using the approach described here. One, the top-ranking system in the CoNLL 2005 shared task is a phrase-based system, described in details in (Punyakanok, Roth, & Yih 2005; 2008). Here we describe a word-based system designed to allow a comparison of the framework presented here with standard CRFs.

We have defined 5 linguistic constraints on the argument structure of the SRL task. For example, the *NoDup* constraint in Table 1, means that verb arguments cannot be duplicated. Due to space limitation, we cannot describe all of the constraints; please see (Roth & Yih 2005) for details.

SRL is modeled as a sequential tagging task. We used a linear chain CRF as our base model (that is, the first term in (1)). In general, our models have two types of features for a sentence:

- For a given word, (F1) captures its local syntactic context (e.g. spelling, POS, . . ., etc., of each word in a window around the target word) and the corresponding state assignment.
- For two given consecutive words, (F2) captures information related to combinations of state assignments of these two words.

Note that these features are not expressive enough to capture the *global* dependencies described by the constraints.

We experimented with three algorithms for training CRF: (1) Maximum posterior likelihood estimation with the L+I protocol, which ignores the constraints during learning. We denote this combination as *CRF-ML*. (2) The Perceptron training algorithm (Collins 2002) with the L+I protocol, denoted *CRF-P*. (3) The Perceptron algorithm using the constraints during learning (IBT protocol), denoted *C-IBT*. In addition, we also tried the (averaged) Voted Perceptron training algorithm which ignores the constraints during learning (L+I protocol), however, unlike (1), (2), and (3), it uses only feature set (F1). We denote this approach as *VP*. Note that in *VP*, the base model assigns values to the output variables independently, relying on constraints to capture the interaction between output variables. Therefore, the feature space of *VP* is smaller than *CRF-ML*.

The results are shown in Table 1. The most interesting result is that although *VP* is the model with weakest feature set, it achieves the best results. With the given amount of

Constraints	CRF-ML	CRF-P	CRF-IBT	VP
None	66.46	69.14	69.14	58.15
+No Dup	67.10	69.74	N/A	64.33
+Candidates	71.78	73.64	N/A	74.17
+Arguments	71.71	73.71	N/A	74.02
+Verb Pos	71.72	73.78	N/A	74.03
+Disallow	71.94	73.91	69.82	74.49
Training Time (hrs)	48	38	145	0.8

Table 1: F_1 scores of different SRL systems using CCM.

labeled data, despite the fact that it takes much longer to train, CRF-IBT performs substantially worse than CRF-P.[3] Although IBT seems to be a reasonable approach, it requires more training examples to achieve good results (Punyakanok *et al.* 2005).

Semi-Supervised Training 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 the models learned from a small training set (Collins & Singer 1999). On the other hand, in the supervised setting, it has been shown that incorporating domain and problem specific prior knowledge as constraints can result in substantial improvements (Roth & Yih 2005).

In this section, we present an algorithm which combines prior knowledge and a semi-supervised learning algorithm (Chang, Ratinov, & Roth 2007). We show that prior knowledge plays a crucial role when the amount of labeled data is limited. The algorithm is based on CCM, since it provides a good platform to combine the learned models and prior knowledge. The algorithm is called the constraint-driven learning algorithm (**CODL**), which uses constraints to guide semi-supervised learning.

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 *constraints* and the learned model to label the instances in the pool, and **(2)** updates the model via the newly labeled data.

$CODL$ is summarized in Figure 2. $CODL$ uses an EM-like procedure. The model is initialized with traditional supervised learning (ignoring the constraints) on a small labeled set (line 1). Given an unlabeled set U, we find the most likely assignment by finding the maximum value of the "scoring" function for CCM, equation (1), which uses both the learned model and the *constraints*. We use beam-search here to approximate the best assignment (line 6, Inference-WithConstraints). In line 8, the model parameters are up-

[3]We did not evaluate the contribution of each constraint for CRF-IBT due to prohibitive time consumption and unsatisfactory performance.

```
Input:
    $N$: learning cycles; $U$: unlabeled dataset; $Tr = \{x, y\}$: labeled training set; $\{\rho_i\}$:
set of penalties; $\{C_i\}$: set of constraints; $\gamma$: balancing parameter with the supervised model;
$learn(Tr)$: supervised learning algorithm;
1.    Initialize $w = w_0 = learn(Tr)$.
2.    $w = w_0$.
3.    For $N$ iterations do:
4.        $T = \phi$
5.        For each $x \in U$
6.            $(x, \hat{y})$ = InferenceWithConstraints($x, w, \{C_i\}, \{\rho_i\}$)
7.            $T = T \cup \{(x, \hat{y})\}$
8.        $w = \gamma w_0 + (1 - \gamma) learn(T)$
```

Figure 2: **CO**nstraint **D**riven **L**earning (CODL).

dated based on the results in line 6. Line 6 is analogous to E-step and line 8 is analogous to M-step of the traditional EM algorithm.

The fact that the constraints are used in the inference procedure (in particular, for generating new training examples) allows us to use a learning algorithm that ignores the constraints (i.e., the L+I protocol discussed previously), which is a lot more efficient. When we learn the algorithm in line 8, we choose not to put constraints in the training procedure, since we have little labeled data, and it is not worthwhile to make the model too complex in this situation.

Line 8 in the algorithm should be further clarified. It is known that traditional semi-supervised training can degrade the learned model's performance. (Nigam *et al.* 2000) has suggested to balance the contribution of labeled and unlabeled data to the parameters. The intuition is that when iteratively estimating the parameters with EM, we restrain the parameters from drifting too far from the supervised model. (Nigam *et al.* 2000) shows that as the amount of training data increases, the weight assigned to the unlabeled data should decrease and proposes simple techniques for tuning the weighting parameter based on a development set. 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.

Case study: Information Extraction

This section reports an application of $CODL$ to information extraction problems, where, given 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.

To illustrate the effectiveness of $CODL$, we consider the task of identifying fields from citations (Peng & McCallum 2006)[4]. The data originally included 500 labeled references, and was later extended with 5,000 unannotated citations collected from papers found on the Internet (Grenager, Klein,

[4]We apply $CODL$ on two applications (Chang, Ratinov, & Roth 2007). However, due to space limitations, we do not report the results of the second problem, where the task is extracting fields from advertisements (Grenager, Klein, & Manning 2005). The data for both problems is available at: http://www.stanford.edu/~grenager/data/unsupie.tgz

(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 .]

Figure 1: Error analysis of an HMM. (a) The correct assignment . (b) The predicted assignment by a HMM trained on 30 examples.

& Manning 2005). Given a citation, the task is to extract the fields that appear in the given reference. See Figure 1. There are 13 possible fields including author, title, location, etc.

To gain insight into how the constraints can guide semi-supervised learning, assume that the sentence shown in Figure 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 a HMM trained on 30 labels. However, if we apply the constraint that states transitions can occur only on punctuation marks, the same HMM model parameters 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, punctuation marks indicate only some of the constraints that can be applied to this problem. Sample constraints used for the citations dataset are:

- Each field must be a consecutive list of words, and can appear at most once in a citation.
- State transitions must occur on punctuation marks.
- The citation can only start with author or editor.
- The words *"proc, journal, proceedings, ACM"* belong to either *JOURNAL* or *BOOKTITLE*.

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.

N	Inf.	**sup.**	**H**	**H&W**	**H&W&C**
5	no I	55.1	60.9	63.6	70.6
	I	66.6	69.0	72.5	76.0
10	no I	64.6	66.8	69.8	76.5
	I	78.1	78.1	81.0	83.4
15	no I	68.7	70.6	73.7	78.6
	I	81.3	81.9	84.1	85.5
20	no I	70.1	72.4	75.0	79.6
	I	81.1	82.4	84.0	86.1
25	no I	72.7	73.2	77.0	81.6
	I	84.3	84.2	86.2	87.4
300	no I	86.1	80.7	87.1	88.2
	I	92.5	89.6	93.4	93.6

Table 2: Extracting fields from citations-results

We trained models on training sets of size varying from 5 to 300, with 1000 unlabeled data samples. To verify our claims we implemented several baselines. The first baseline is the supervised learning protocol denoted by **sup**. The second baseline was a traditional **top-1 Hard EM**, which is also known as truncated EM[5] (denoted by **H** for Hard). In the third baseline, denoted **H&W**, we balanced the weight of the supervised and unsupervised models as described in line 8 of Figure 2. We compare these baselines to our proposed protocol, **H&W&C**, where we added the constraints to guide **H&W**.[6]

The experimental results are given in Table 2. As hypothesized, hard EM sometimes degrades the performance. Indeed, with 300 labeled examples in the citations domain, the performance decreases from 86.1 to 80.7. The usefulness of injecting constraints in semi-supervised learning is exhibited in the rightmost column: using constraints, **H&W&C** improves the performance over **H&W** quite significantly.

To examine the contribution of using constraints to the learning stage and the testing stage, two separate results are presented: testing with constraints (denoted *I* for inference) and without constraints (*no I*). The *I* results are consistently better. It is also clear from Table 2 that testing with constraints is more important than using constraints during learning, especially when the labeled data size is large. This confirms results reported for the supervised learning case in (Punyakanok *et al.* 2005; Roth & Yih 2005). However, our proposed algorithm **H&W&C** for *training with constraints* is critical when the amount labeled data is small.

Conclusion

In this paper, we summarize and provide a unified view of a framework aimed at facilitating decision making with respect to multiple interdependent variables the values of which are determined by learned probabilistic models. We proposed a Constrained Conditional Model, 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. Importantly, this framework provides a principled way to incorporate expressive background knowledge into the decision process, possibly, knowledge available only at decision time, without unnecessarily modifying the model. It also provides way to combine conditional models, learned independently in different situations, along with declarative information to support coherent global decisions. We have described several very successful applications of this framework in the NLP

[5]We also experimented with (soft) EM without constraints, but the results were generally worse.

[6]We performed a crude tuning on a held-out set for all the experiments on the citations domain, resulting in the following parameter settings: $\gamma = 0.1$, and the constraints violation penalty $\rho = log(10^{-4})$ (based on the intuition that the penalty corresponds to the log-likelihood of the constraint being violated).

domain, and discussed some important issues that pertain to training and testing these models. We believe that the use of the modeling language developed for CCMs, which facilitates programming systems that use conditional models with declarative constraints (Rizzolo & Roth 2007) will allow researchers and application designers to further study this framework.

Acknowledgments

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., and Lapata, M. 2006. Aggregation via Set Partitioning for Natural Language Generation. In *Proc. of HLT/NAACL*.

Chang, M.; Ratinov, L.; and Roth, D. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL*.

Collins, M., and Singer, Y. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

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

Dechter, R., and Mateescu, R. 2004. Mixtures of deterministic-probabilistic networks and their AND/OR search space. In *Proceedings of AUAI*, 120–129. Arlington, Virginia, United States: AUAI Press.

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

Koomen, P.; Punyakanok, V.; Roth, D.; and Yih, W. 2005. Generalized inference with multiple semantic role labeling systems (shared task paper). In *Proc. of the CoNLL*.

Marciniak, T., and Strube, M. 2005. Beyond the Pipeline: Discrete Optimization in NLP. In *Proc. of the CoNLL*.

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

Palmer, M.; Gildea, D.; and Kingsbury, P. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics* 31(1):71–106.

Peng, F., and McCallum, A. 2006. Information extraction from research papers using conditional random fields. *Inf. Process. Manage.* 42(4):963–979.

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

Punyakanok, V.; Roth, D.; and Yih, W. 2005. The Necessity of Syntactic Parsing for Semantic Role Labeling. In *Proc. of IJCAI*.

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

Rizzolo, N., and Roth, D. 2007. Modeling Discriminative Global Inference. In *Proc. of the ICSC*, 597–604. IEEE.

Roth, D., and Yih, W. 2004. A linear programming formulation for global inference in natural language tasks. In Ng, H. T., and Riloff, E., eds., *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, 1–8. Association for Computational Linguistics.

Roth, D., and Yih, W. 2005. Integer Linear Programming Inference for Conditional Random Fields. In *Proc. of ICML*.

Roth, D., and Yih, W. 2007. Global inference for entity and relation identification via a linear programming formulation. In Getoor, L., and Taskar, B., eds., *Introduction to Statistical Relational Learning*. MIT Press.

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