Guiding Semi-Supervision with Constraint-Driven Learning

Ming-Wei Chang **Lev Ratinov** **Dan Roth**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{mchang21, ratinov2, danr}@uiuc.edu

Abstract

Over the last few years, two of the main research directions in machine learning of natural language processing have been the study of semi-supervised learning algorithms as a way to train classifiers when the labeled data is scarce, and the study of ways to exploit knowledge and global information in structured learning tasks. In this paper, we suggest a method for incorporating domain knowledge in semi-supervised learning algorithms. Our novel framework unifies and can exploit several kinds of *task specific constraints*. The experimental results presented in the information extraction domain demonstrate that applying constraints helps the model to generate better feedback during learning, and hence the framework allows for high performance learning with significantly less training data than was possible before on these tasks.

1 Introduction

Natural Language Processing (NLP) systems typically require large amounts of knowledge to achieve good performance. 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 and Singer, 1999; Thelen and Riloff, 2002). The hope is that semi-supervised or even unsupervised approaches, when given enough knowledge about the *structure* of the problem, will be competitive with the supervised models trained on large training sets. However, in the general case, semi-supervised approaches give mixed results, and sometimes even degrade the model performance (Nigam et al., 2000). In many cases, improving semi-supervised models was done by *seeding* these models with domain information taken from dictionaries or ontology (Cohen and Sarawagi, 2004; Collins and Singer, 1999; Haghighi and Klein, 2006; Thelen and Riloff, 2002). On the other hand, in the supervised setting, it has been shown that incorporating domain and problem specific structured information can result in substantial improvements (Toutanova et al., 2005; Roth and Yih, 2005).

This paper proposes a novel constraints-based learning protocol for guiding semi-supervised learning. We develop a formalism for constraints-based learning that unifies several kinds of constraints: unary, dictionary based and n-ary constraints, which encode structural information and interdependencies among possible labels. One advantage of our formalism is that it allows capturing different levels of constraint violation. Our protocol can be used in the presence of any learning model, including those that acquire additional statistical constraints from observed data while learning (see Section 5. In the experimental part of this paper we use HMMs as the underlying model, and exhibit significant reduction in the number of training examples required in two information extraction problems.

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

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 280–287,
Prague, Czech Republic, June 2007.

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.
(2) Updates the model by newly labeled data.

This way, we can generate *better* "training" examples during the semi-supervised learning process. The core of our approach, (1), is described in Section 5. The task is described in Section 3 and the Experimental study in Section 6. It is shown there that the improvement on the training examples via the constraints indeed boosts the learned model and the proposed method significantly outperforms the traditional semi-supervised framework.

2 Related Work

In the semi-supervised domain there are two main approaches for injecting domain specific knowledge. One is using the prior knowledge to accurately tailor the generative model so that it captures the domain structure. For example, (Grenager et al., 2005) proposes *Diagonal Transition Models* for sequential labeling tasks where neighboring words tend to have the same labels. This is done by constraining the HMM transition matrix, which can be done also for other models, such as CRF. However (Roth and Yih, 2005) showed that reasoning with more expressive, non-sequential constraints can improve the performance for the supervised protocol.

A second approach has been to use a small high-accuracy set of labeled tokens as a way to seed and bootstrap the semi-supervised learning. This was used, for example, by (Thelen and Riloff, 2002; Collins and Singer, 1999) in information extraction, and by (Smith and Eisner, 2005) in POS tagging. (Haghighi and Klein, 2006) extends the dictionary-based approach to sequential labeling tasks by propagating the information given in the seeds with contextual word similarity. This follows a conceptually similar approach by (Cohen and Sarawagi, 2004) that uses a large named-entity dictionary, where the similarity between the candidate named-entity and its matching prototype in the dictionary is encoded as a feature in a supervised classifier.

In our framework, dictionary lookup approaches are viewed as unary constraints on the output states. We extend these kinds of constraints and allow for more general, n-ary constraints.

In the supervised learning setting it has been established that incorporating global information can significantly improve performance on several NLP tasks, including information extraction and semantic role labeling. (Punyakanok et al., 2005; Toutanova et al., 2005; Roth and Yih, 2005). Our formalism is most related to this last work. But, we develop a semi-supervised learning protocol based on this formalism. We also make use of *soft* constraints and, furthermore, extend the notion of soft constraints to account for multiple levels of constraints' violation. Conceptually, although not technically, the most related work to ours is (Shen et al., 2005) that, in a somewhat ad-hoc manner uses soft constraints to guide an unsupervised model that was crafted for mention tracking. To the best of our knowledge, we are the first to suggest a general semi-supervised protocol that is driven by soft constraints.

We propose learning with constraints - a framework that combines the approaches described above in a unified and intuitive way.

3 Tasks, Examples and Datasets

In Section 4 we will develop a general framework for semi-supervised learning with constraints. However, it is useful to illustrate the ideas on concrete problems. Therefore, in this section, we give a brief introduction to the two domains on which we tested our algorithms. We study two information extraction problems in each of which, given text, a set of pre-defined 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.[1]

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

[1]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 a HMM model. The labels are annotated by underline and are to the right of each open bracket. The correct assignment was shown in (a). While the predicted label assignment (b) is generally coherent, some constraints are violated. Most obviously, punctuation marks are ignored as cues for state transitions. The constraint "Fields cannot end with stop words (such as "the")" may be also good.

fields that appear in the given reference. See Fig. 1. There are 13 possible fields including author, title, location, etc.

To gain an insight to 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 state transition 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, the 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 1. 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 seedwords due to difference in preprocessing.

4 Notation and Definitions

Consider a structured classification problem, where given an input sequence $x = (x_1, \dots, x_N)$, the task is to find the best assignment to the output variables $y = (y_1, \dots, y_M)$. We denote $\mathcal{X}$ to be the space of the possible input sequences and $\mathcal{Y}$ to be the set of possible output sequences.

We define a structured output classifier as a function $h : \mathcal{X} \to \mathcal{Y}$ that uses a global scoring function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ to assign scores to each possible input/output pair. Given an input x, a desired function f will assign the correct output y the highest score among all the possible outputs. The global scoring function is often decomposed as a weighted sum of feature functions,

$$f(x, y) = \sum_{i=1}^{M} \lambda_i f_i(x, y) = \boldsymbol{\lambda} \cdot F(x, y).$$

This decomposition applies both to discriminative linear models and to generative models such as HMMs and CRFs, in which case the linear sum corresponds to log likelihood assigned to the input/output pair by the model (for details see (Roth, 1999) for the classification case and (Collins, 2002) for the structured case). Even when not dictated by the model, the feature functions $f_i(x, y)$ used are local to allow inference tractability. Local feature function can capture some context for each input or output variable, yet it is very limited to allow dynamic programming decoding during inference.

Now, consider a scenario where we have a set of constraints $C_1, \dots, C_K$. We define a constraint $C : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ as a function that indicates whether the input/output sequence violates some desired properties. When the constraints are hard, the solution is given by

$$\operatorname*{argmax}_{y \in 1_{C(x)}} \boldsymbol{\lambda} \cdot F(x, y),$$

(a)-Citations

1) Each field must be a consecutive list of words, and can appear at most once in a citation.
2) State transitions must occur on punctuation marks.
3) The citation can only start with author or editor.
4) The words *pp., pages* correspond to *PAGE*.
5) Four digits starting with 20xx and 19xx are *DATE*.
6) Quotations can appear only in titles.
7) The words *note, submitted, appear* are *NOTE*.
8) The words *CA, Australia, NY* are *LOCATION*.
9) The words *tech, technical* are *TECH_REPORT*.
10) The words *proc, journal, proceedings, ACM* are *JOURNAL* or *BOOKTITLE*.
11) The words *ed, editors* correspond to *EDITOR*.

(b)-Advertisements

1) State transitions can occur only on punctuation marks or the newline symbol.
2) Each field must be at least 3 words long.
3) The words *laundry, kitchen, parking* are *FEATURES*.
4) The words *sq, ft, bdrm* are *SIZE*.
5) The word *$, *MONEY** are *RENT*.
6) The words *close, near, shopping* are *NEIGHBORHOOD*.
7) The words *laundry kitchen, parking* are *FEATURES*.
8) The (normalized) words *phone, email* are *CONTACT*.
9) The words *immediately, begin, cheaper* are *AVAILABLE*.
10) The words *roommates, respectful, drama* are *ROOMMATES*.
11) The words *smoking, dogs, cats* are *RESTRICTIONS*.
12) The word *http, image, link* are *PHOTOS*.
13) The words *address, carlmont, st, cross* are *ADDRESS*.
14) The words *utilities, pays, electricity* are *UTILITIES*.

Table 1: The list of constraints for extracting fields from citations and advertisements. Some constraints (represented in the first block of each domain) are global and are relatively difficult to inject into traditional models. While all the constraints hold for the vast majority of the data, some of them are violated by some correct labeled assignments.

where $1_{C(x)}$ is a subset of $\mathcal{Y}$ for which all C_i assign the value 1 for the given (x, y).

When the constraints are soft, we want to incur some penalty for their violation. Moreover, we want to incorporate into our cost function a measure for the *amount* of violation incurred by violating the constraint. A generic way to capture this intuition is to introduce a distance function $d(y, 1_{C_i(x)})$ between the space of outputs that respect the constraint, $1_{C_i(x)}$, and the given output sequence y. One possible way to implement this distance function is as the minimal Hamming distance to a sequence that respects the constraint C_i, that is: $d(y, 1_{C_i(x)}) = \min_{(y' \in 1_{C(x)})} H(y, y')$. If the penalty for violating the soft constraint C_i is ρ_i, we write the score function as:

$$\underset{y}{\operatorname{argmax}}\, \boldsymbol{\lambda} \cdot F(x, y) - \sum_{i=1}^{K} \rho_i d(y, 1_{C_i(x)}) \qquad (1)$$

We refer to $d(y, 1_{C(x)})$ as the *valuation* of the constraint C on (x, y). The intuition behind (1) is as follows. Instead of merely maximizing the model's likelihood, we also want to bias the model using some knowledge. The first term of (1) is used to learn from data. The second term biases the mode by using the knowledge encoded in the constraints. Note that we do not normalize our objective function to be a true probability distribution.

5 Learning and Inference with Constraints

In this section we present a new constraint-driven learning algorithm (**CODL**) for using constraints to guide semi-supervised learning. The task is to learn the parameter vector $\boldsymbol{\lambda}$ by using the new objective function (1). While our formulation allows us to train also the coefficients of the constraints valuation, ρ_i, we choose not to do it, since we view this as a way to bias (or enforce) the prior knowledge into the learned model, rather than allowing the data to brush it away. Our experiments demonstrate that the proposed approach is robust to inaccurate approximation of the prior knowledge (assigning the same penalty to all the ρ_i).

We note that in the presence of constraints, the inference procedure (for finding the output y that maximizes the cost function) is usually done with search techniques (rather than Viterbi decoding, see (Toutanova et al., 2005; Roth and Yih, 2005) for a discussion), we chose beamsearch decoding.

The semi-supervised learning with constraints is done with an EM-like procedure. We initialize the model with traditional supervised learning (ignoring the constraints) on a small labeled set. Given an unlabeled set U, in the estimation step, the traditional EM algorithm assigns a distribution over labeled assignments $\mathcal{Y}$ of each $x \in U$, and in the maximization step, the set of model parameters is learned from the distributions assigned in the estimation step.

However, in the presence of constraints, assigning the complete distributions in the estimation step is infeasible since the constraints reshape the distribution in an arbitrary way. As in existing methods for training a model by maximizing a linear cost function (maximize likelihood or discriminative maximization), the distribution over $\mathcal{Y}$ is represented as the set of scores assigned to it; rather than considering the score assigned to *all* $y's$, we truncate the distribution to the top K assignments as returned by the search. Given a set of K top assignments $y^1, \ldots, y^K$, we approximate the estimation step by assigning uniform probability to the top K candidates, and zero to the other output sequences. We denote this algorithm *top-K hard EM*. In this paper, we use beamsearch to generate K candidates according to (1).

Our training algorithm is summarized in Figure 2. Several things about the algorithm should be clarified: the Top-K-Inference procedure in line 7, the learning procedure in line 9, and the new parameter estimation in line 9.

The Top-K-Inference is a procedure that returns the K labeled assignments that maximize the new objective function (1). In our case we used the top-K elements in the beam, but this could be applied to any other inference procedure. 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, which is a lot more efficient (although algorithms that do take the constraints into account can be used too). We used maximum likelihood estimation of $\boldsymbol{\lambda}$ but, in general, perceptron or quasi-Newton can also be used.

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 disallow the parameters to drift too far from the supervised model. The parameter re-estimation in line 9, uses a similar intuition, but instead of weighting data instances, we introduced a smoothing parameter γ which controls the convex combination of models induced by the labeled and the unlabeled data. Unlike the technique mentioned above which focuses on naive Bayes, our method allows us to weight linear models generated by different learning algorithms.

Another way to look the algorithm is from the self-training perspective (McClosky et al., 2006). Similarly to self-training, we use the current model to generate new training examples from the unlabeled set. However, there are two important differences. One is that in self-training, once an unlabeled sample was labeled, it is never labeled again. In our case all the samples are relabeled in each iteration. In self-training it is often the case that only high-confidence samples are added to the labeled data pool. While we include all the samples in the training pool, we could also limit ourselves to the high-confidence samples. The second difference is that each unlabeled example generates K labeled instances. The case of one iteration of *top-1 hard EM* is equivalent to self training, where all the unlabeled samples are added to the labeled pool.

Input:
- $Cycles$: learning cycles
- $Tr = \{x, y\}$: labeled training set.
- U: unlabeled dataset
- F: set of feature functions.
- $\{\rho_i\}$: set of penalties.
- $\{C_i\}$: set of constraints.
- γ: balancing parameter with the supervised model.
- $learn(Tr, F)$: supervised learning algorithm
- Top-K-Inference:
  returns top-K labeled scored by the cost function (1)

CODL:

1. Initialize $\boldsymbol{\lambda}_0 = learn(Tr, F)$.
2. $\boldsymbol{\lambda} = \boldsymbol{\lambda}_0$.
3. For $Cycles$ iterations do:
4. $\quad T = \phi$
5. $\quad$ For each $x \in U$
6. $\quad\quad \{(x, y^1), \ldots, (x, y^K)\} =$
7. $\quad\quad\quad$ Top-K-Inference$(x, \boldsymbol{\lambda}, F, \{C_i\}, \{\rho_i\})$
8. $\quad\quad T = T \cup \{(x, y^1), \ldots, (x, y^K)\}$
9. $\quad \boldsymbol{\lambda} = \gamma \boldsymbol{\lambda}_0 + (1 - \gamma) learn(T, F)$

Figure 2: **CO**nstraint **D**riven **L**earning (CODL). In Top-K-Inference, we use beamsearch to find the K-best solution according to Eq. (1).

There are several possible benefits to using $K > 1$ samples. (1) It effectively increases the training set by a factor of K (albeit by somewhat noisy examples). In the structured scenario, each of the top-K assignments is likely to have some good components so generating top-K assignments helps leveraging the noise. (2) Given an assignment that does not satisfy some constraints, using top-K allows for multiple ways to correct it. For example, consider the output 11101000 with the constraint that it should belong to the language 1*0*. If the two top scoring corrections are 11111000 and 11100000, considering only one of those can negatively bias the model.

6 Experiments and Results

In this section, we present empirical results of our algorithms on two domains: *citations* and *advertisements*. Both problems are modeled with a simple token-based HMM. We stress that token-based HMM cannot represent many of our constraints. The function $d(y, 1_{C(x)})$ used is an approximation of a Hamming distance function, discussed in Section 7. For both domains, and all the experiments, γ was set to 0.1. The constraints violation penalty ρ is set to $-\log 10^{-4}$ and $-\log 10^{-1}$ for citations and advertisements, resp.[2] Note that all constraints share the same penalty. The number of semi-supervised training cycles (line 3 of Figure 2) was set to 5. The constraints for the two domains are listed in Table 1.

We trained models on training sets of size varying from 5 to 300 for the citations and from 5 to 100 for the advertisements. Additionally, in all the semi-supervised experiments, 1000 unlabeled examples are used. We report token-based[3] accuracy on 100 held-out examples (which do not overlap neither with the training nor with the unlabeled data). We ran 5 experiments in each setting, randomly choosing the training set. The results reported below are the averages over these 5 runs.

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** also known as truncated EM[4] (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 9 of Figure 2. We compare these baselines to our proposed protocol, **H&W&C**, where we added the constraints to guide the **H&W** protocol. We experimented with two flavors of the algorithm: the top-1 and the top-K version. In the top-K version, the algorithm uses K-best predictions (K=50) for each instance in order to update the model as described in Figure 2.

The experimental results for both domains are in given Table 2. As hypothesized, hard EM sometimes degrade 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 two right most columns: using constraints **H&W&C** improves the performance over **H&W** quite significantly.

We carefully examined the contribution of using constraints to the learning stage and the testing stage, and two separate results are presented: testing with constraints (denoted *I* for inference) and without constraints (*no I*). The *I* results are consistently better. And, it is also clear from Table 2, that using constraints in training always improves

(a)- Citations

N	Inf.	**sup.**	**H**	**H&W**	**H&W&C** (Top-1)	**H&W&C** (Top-K)
5	no I	55.1	60.9	63.6	70.6	71.0
	I	66.6	69.0	72.5	76.0	77.8
10	no I	64.6	66.8	69.8	76.5	76.7
	I	78.1	78.1	81.0	83.4	83.8
15	no I	68.7	70.6	73.7	78.6	79.4
	I	81.3	81.9	84.1	85.5	86.2
20	no I	70.1	72.4	75.0	79.6	79.4
	I	81.1	82.4	84.0	86.1	86.1
25	no I	72.7	73.2	77.0	81.6	82.0
	I	84.3	84.2	86.2	87.4	87.6
300	no I	86.1	80.7	87.1	88.2	88.2
	I	92.5	89.6	93.4	93.6	93.5

(b)-Advertisements

N	Inf.	**sup.**	**H**	**H&W**	**H&W&C** (Top-1)	**H&W&C** (Top-K)
5	no I	55.2	61.8	60.5	66.0	66.0
	I	59.4	65.2	63.6	69.3	69.6
10	no I	61.6	69.2	67.0	70.8	70.9
	I	66.6	73.2	71.6	74.7	74.7
15	no I	66.3	71.7	70.1	73.0	73.0
	I	70.4	75.6	74.5	76.6	76.9
20	no I	68.1	72.8	72.0	74.5	74.6
	I	71.9	76.7	75.7	77.9	78.1
25	no I	70.0	73.8	73.0	74.9	74.8
	I	73.7	77.7	76.6	78.4	78.5
100	no I	76.3	76.2	77.6	78.5	78.6
	I	80.4	80.5	81.2	81.8	81.7

Table 2: Experimental results for extracting fields from citations and advertisements. N is the number of labeled samples. **H** is the traditional hard-EM and **H&W** weighs labeled and unlabeled data as mentioned in Sec. 5. Our proposed model is **H&W&C**, which uses constraints in the learning procedure. **I** refers to using constraints during inference at evaluation time. Note that adding constraints improves the accuracy during both learning and inference.

[2]The guiding intuition is that $\boldsymbol{\lambda} F(x, y)$ corresponds to a log-likelihood of a HMM model and ρ to a crude estimation of the log probability that a constraint does not hold. ρ was tuned on a development set and kept fixed in all experiments.

[3]Each token (word or punctuation mark) is assigned a state.

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

the model and the amount of improvement depends on the amount of labeled data.

Figure 3 compares two protocols on the advertisements domain: **H&W+I**, where we first run the **H&W** protocol and then apply the constraints during testing stage, and **H&W&C+I**, which uses constraints to guide the model during learning and uses it also in testing. Although injecting constraints in the learning process helps, 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 and Yih, 2005). However, as shown, our proposed algorithm **H&W&C** for *training with constraints* is critical when the amount labeled data is small.

Figure 4 further strengthens this point. In the citations domain, **H&W&C+I** achieves with 20 labeled samples similar performance to the supervised version without constraints with 300 labeled samples.

(Grenager et al., 2005) and (Haghighi and Klein, 2006) also report results for semi-supervised learning for these domains. However, due to different preprocessing, the comparison is not straightforward. For the citation domain, when 20 labeled and 300 unlabeled samples are available, (Grenager et al., 2005) observed an increase from 65.2% to 71.3%. Our improvement is from 70.1% to 79.4%. For the advertisement domain, they observed no improvement, while our model improves from 68.1% to 74.6% with 20 labeled samples. Moreover, we successfully use out-of-domain data (web data) to improve our model, while they report that this data did not improve their unsupervised model.

(Haghighi and Klein, 2006) also worked on one of our data sets. Their underlying model, Markov Random Fields, allows more expressive features. Nevertheless, when they use only unary constraints they get 53.75%. When they use their final model, along with a mechanism for extending the prototypes to other tokens, they get results that are comparable to our model with 10 labeled examples. Additionally, in their framework, it is not clear how to use small amounts of labeled data when available. Our model outperforms theirs once we add 10 more examples.

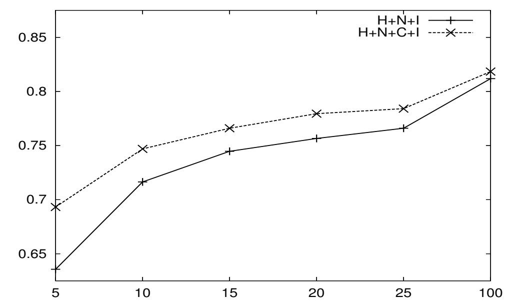

Figure 3: Comparison between **H&W+I** and **H&W&C+I** on the *advertisements* domain. When there is a lot of labeled data, inference with constraints is more important than using constraints during learning. However, it is important to train with constraints when the amount of labeled data is small.

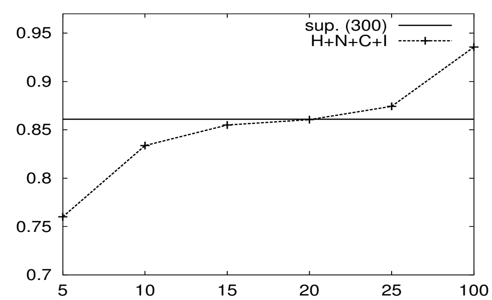

Figure 4: With 20 labeled citations, our algorithm performs competitively to the supervised version trained on 300 samples.

7 Soft Constraints

This section discusses the importance of using soft constraints rather than hard constraints, the choice of Hamming distance for $d(y, 1_{C(x)})$ and how we approximate it. We use two constraints to illustrate the ideas. *(C_1):* "state transitions can only occur on punctuation marks or newlines", and *(C_2):* "the field *TITLE* must appear".

First, we claim that defining $d(y, 1_{C(x)})$ to be the Hamming distance is superior to using a binary value, $d(y, 1_{C(x)}) = 0$ if $y \in 1_{C(x)}$ and 1 otherwise. Consider, for example, the constraint C_1 in the advertisements domain. While the vast majority of the instances satisfy the constraint, some violate it in more than one place. Therefore, once the binary distance is set to 1, the algorithm looses the ability to discriminate constraint violations in other locations of the same instance. This may hurt the performance in both the inference and the learning stage.

Computing the Hamming distance exactly can be a computationally hard problem. Furthermore, it is unreasonable to implement the exact computation for each constraint. Therefore, we implemented a generic approximation for the hamming distance assuming only that we are given a boolean function $\phi_C(y_N)$ that returns whether labeling the token x_N with state y_N violates constraint with respect to an already labeled sequence $(x_1, \ldots, x_{N-1}, y_1, \ldots, y_{N-1})$. Then $d(y, 1_{C(x)}) = \sum_{i=1}^{N} \phi_C(y_i)$. For example, consider the prefix x_1, x_2, x_3, x_4, which contains no punctuation or newlines and was labeled $AUTH, AUTH, DATE, DATE$. This labeling violates C_1, the minimal hamming distance is 2, and our approximation gives 1, (since there is only one transition that violates the constraint.)

For constraints which cannot be validated based on prefix information, our approximation resorts to binary violation count. For instance, the constraint C_2 cannot be implemented with prefix information when the assignment is not complete. Otherwise, it would mean that the field TITLE should appear as early as possible in the assignment.

While (Roth and Yih, 2005) showed the significance of using hard constraints, our experiments show that using soft constraints is a superior option. For example, in the advertisements domain, C_1 holds for the large majority of the gold-labeled instances, but is sometimes violated. In supervised training with 100 labeled examples on this domain, **sup** gave 76.3% accuracy. When the constraint violation penalty ρ was *infinity* (equivalent to hard constraint), the accuracy improved to 78.7%, but when the penalty was set to $-log(0.1)$, the accuracy of the model jumped to 80.6%.

8 Conclusions and Future Work

We proposed to use constraints as a way to guide semi-supervised learning. The framework developed is general both in terms of the representation and expressiveness of the constraints, and in terms of the underlying model being learned – HMM in the current implementation. Moreover, our framework is a useful tool when the domain knowledge cannot be expressed by the model.

The results show that constraints improve not only the performance of the final inference stage but also propagate useful information during the semi-supervised learning process and that training with the constraints is especially significant when the number of labeled training data is small.

Acknowledgments: This work is supported by NSF SoD-HCER-0613885 and by a grant from Boeing. Part of this work was done while Dan Roth visited the Technion, Israel, supported by a Lady Davis Fellowship.

References

W. Cohen and S. Sarawagi. 2004. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In *Proc. of the ACM SIGKDD*.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proc. of EMNLP*.

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

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

A. Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. In *Proc. of HTL-NAACL*.

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

D. McClosky, E. Charniak, and M. Johnson. 2006. Effective self-training for parsing. In *Proceedings of HLT-NAACL*.

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

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

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

D. Roth. 1999. Learning in natural language. In *Proc. of IJCAI*, pages 898–904.

W. Shen, X. Li, and A. Doan. 2005. Constraint-based entity matching. In *Proc. of AAAI)*.

N. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of the Annual Meeting of the ACL*.

M. Thelen and E. Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proc. of EMNLP*.

K. Toutanova, A. Haghighi, and C. D. Manning. 2005. Joint learning improves semantic role labeling. In *Proc. of the Annual Meeting of the ACL*.