Cross-Task Knowledge-Constrained Self Training

Hal Daumé III
School of Computing
University of Utah
Salt Lake City, UT 84112
me@hal3.name

Abstract

We present an algorithmic framework for learning multiple related tasks. Our framework exploits a form of prior knowledge that relates the output spaces of these tasks. We present PAC learning results that analyze the conditions under which such learning is possible. We present results on learning a shallow parser and named-entity recognition system that exploits our framework, showing consistent improvements over baseline methods.

1 Introduction

When two NLP systems are run on the same data, we expect certain constraints to hold between their outputs. This is a form of prior knowledge. We propose a self-training framework that uses such information to significantly boost the performance of one of the systems. The key idea is to perform self-training *only* on outputs that obey the constraints.

Our motivating example in this paper is the task pair: named entity recognition (NER) and shallow parsing (aka syntactic chunking). Consider a hidden sentence with known POS and syntactic structure below. Further consider four potential NER sequences for this sentence.

POS:	NNP NNP VBD TO NNP NN
Chunk:	[- NP -][- VP -][-PP-][- NP -][-NP-]
NER1:	[- Per -][- O -][-Org-][- 0 -]
NER2:	[- Per -][- O -][- O -][- O -][- O -]
NER3:	[- Per -][- O -][- O -][- Org -]
NER4:	[- Per -][- O -][- O -][-Org-][- O -]

Without ever seeing the actual sentence, can we guess which NER sequence is correct? NER1 seems wrong because we feel like named entities should not be part of verb phrases. NER2 seems wrong because there is an NNP[1] (proper noun) that is not part of a named entity (word 5). NER3 is amiss because we feel it is unlikely that a *single* name should span more than one NP (last two words). NER4 has none of these problems and seems quite reasonable. In fact, for the hidden sentence, NER4 is correct[2].

The remainder of this paper deals with the problem of formulating such prior knowledge into a workable system. There are similarities between our proposed model and both self-training and co-training; background is given in Section 2. We present a formal model for our approach and perform a simple, yet informative, analysis (Section 3). This analysis allows us to define what good and bad constraints are. Throughout, we use a running example of NER using hidden Markov models to show the efficacy of the method and the relationship between the theory and the implementation. Finally, we present full-blown results on seven different NER data sets (one from CoNLL, six from ACE), comparing our method to several competitive baselines (Section 4). We see that for many of these data sets, less than one hundred labeled NER sentences are required to get state-of-the-art performance, using a discriminative sequence labeling algorithm (Daumé III and Marcu, 2005).

2 Background

Self-training works by learning a model on a small amount of labeled data. This model is then evalu-

[1]When we refer to NNP, we also include NNPS.

[2]The sentence is: "George Bush spoke to Congress today"

ated on a large amount of unlabeled data. Its predictions are assumed to be correct, and it is retrained on the unlabeled data according to its own predictions. Although there is little theoretical support for self-training, it is relatively popular in the natural language processing community. Its success stories range from parsing (McClosky et al., 2006) to machine translation (Ueffing, 2006). In some cases, self-training takes into account *model confidence*.

Co-training (Yarowsky, 1995; Blum and Mitchell, 1998) is related to self-training, in that an algorithm is trained on its own predictions. Where it differs is that co-training learns two *separate* models (which are typically assumed to be independent; for instance by training with disjoint feature sets). These models are both applied to a large repository of unlabeled data. Examples on which these two models *agree* are extracted and treated as labeled for a new round of training. In practice, one often also uses a notion of model confidence and only extracts agreed-upon examples for which both models are confident. The original, and simplest analysis of co-training is due to Blum and Mitchell (1998). It does not take into account confidence (to do so requires a *significantly* more detailed analysis (Dasgupta et al., 2001)), but is useful for understanding the process.

3 Model

We define a formal PAC-style (Valiant, 1994) model that we call the "hints model"[3]. We have an instance space $\mathcal{X}$ and *two* output spaces $\mathcal{Y}_1$ and $\mathcal{Y}_2$. We assume two concept classes $\mathcal{C}_1$ and $\mathcal{C}_2$ for each output space respectively. Let $\mathcal{D}$ be a distribution over $\mathcal{X}$, and $f_1 \in \mathcal{C}_1$ (resp., $f_2 \in \mathcal{C}_2$) be target functions. The goal, of course, is to use a finite sample of examples drawn from $\mathcal{D}$ (and labeled—perhaps with noise—by f_1 and f_2) to "learn" $h_1 \in \mathcal{C}_1$ and $h_2 \in \mathcal{C}_2$, which are good approximations to f_1 and f_2.

So far we have not made use of any notion of constraints. Our expectation is that if we constrain h_1 and h_2 to *agree* (vis-a-vis the example in the Introduction), then we should need fewer labeled examples to learn either. (The agreement should "shrink" the size of the corresponding hypothesis spaces.) To formalize this, let $\chi : \mathcal{Y}_1 \times \mathcal{Y}_2 \to \{0, 1\}$ be a *constraint function.* We say that two outputs $y_1 \in \mathcal{Y}_1$ and $y_2 \in \mathcal{Y}_2$ are *compatible* if $\chi(y_1, y_2) = 1$. We need to assume that χ is correct:

Definition 1. *We say that χ is* correct *with respect to $\mathcal{D}, f_1, f_2$ if whenever x has non-zero probability under $\mathcal{D}$, then $\chi(f_1(x), f_2(x)) = 1$.*

RUNNING EXAMPLE

In our example, $\mathcal{Y}_1$ is the space of all POS/chunk sequences and $\mathcal{Y}_2$ is the space of all NER sequences. We assume that $\mathcal{C}_1$ and $\mathcal{C}_2$ are both represented by HMMs over the appropriate state spaces. The functions we are trying to learn are f_1, the "true" POS/chunk labeler and f_2, the "true" NER labeler. (Note that we assume $f_1 \in \mathcal{C}_1$, which is obviously not true for language.)

Our constraint function χ will require the following for agreement: (1) any NNP must be part of a named entity; (2) any named entity must be a subsequence of a noun phrase. This is precisely the set of constraints discussed in the introduction.

The question is: given this additional source of knowledge (i.e., χ), has the learning problem become easier? That is, can we learn f_2 (and/or f_1) using significantly fewer labeled examples than if we did not have χ? Moreover, we have assumed that χ is *correct*, but is this enough? Intuitively, no: a function χ that returns 1 regardless of its inputs is clearly not useful. Given this, what other constraints must be placed on χ. We address these questions in Sections 3.3. However, first we define our algorithm.

3.1 One-sided Learning with Hints

We begin by considering a simplified version of the "learning with hints" problem. Suppose that all we care about is learning f_2. We have a small amount of data labeled by f_2 (call this D) and a *large* amount of data labeled by f_1 (call this D^{unlab}–"unlab" because as far as f_2 is concerned, it is unlabeled).

RUNNING EXAMPLE

In our example, this means that we have a small amount of labeled NER data and a large amount of labeled POS/chunk data. We use 3500 sentences from CoNLL (Tjong Kim Sang and De Meulder, 2003) as the NER data and section 20-23 of the WSJ (Marcus et al., 1993; Ramshaw and Marcus, 1995) as the POS/chunk data (8936 sentences). We are *only* interested in learning to do NER. Details of the exact HMM setup are in Section 4.2.

[3]The name comes from thinking of our knowledge-based constraints as "hints" to a learner as to what it should do.

We call the following algorithm "One-Sided Learning with Hints," since it aims only to learn f_2:

```
1: Learn $h_2$ directly on $D$
2: For each example $(x, y_1) \in D^{\text{unlab}}$
3:     Compute $y_2 = h_2(x)$
4:     If $\chi(y_1, y_2)$, add $(x, y_2)$ to $D$
5: Relearn $h_2$ on the (augmented) $D$
6: Go to (2) if desired
```

RUNNING EXAMPLE

In step 1, we train an NER HMM on CoNLL. On test data, this model achieves an F-score of 50.8. In step 2, we run this HMM on all the WSJ data, and extract 3145 compatible examples. In step 3, we retrain the HMM; the F-score rises to 58.9.

3.2 Two-sided Learning with Hints

In the two-sided version, we assume that we have a small amount of data labeled by f_1 (call this D_1), a small amount of data labeled by f_2 (call this D_2) and a *large* amount of unlabeled data (call this D^{unlab}). The algorithm we propose for learning hypotheses for both tasks is below:

```
1: Learn $h_1$ on $D_1$ and $h_2$ on $D_2$.
2: For each example $x \in D^{\text{unlab}}$:
3:     Compute $y_1 = h_1(x)$ and $y_2 = h_2(x)$
4:     If $\chi(y_1, y_2)$ add $(x, y_1)$ to $D_1$, $(x, y_2)$ to $D_2$
5: Relearn $h_1$ on $D_1$ and $h_2$ on $D_2$.
6: Go to (2) if desired
```

RUNNING EXAMPLE

We use 3500 examples from NER and 1000 from WSJ. We use the remaining 18447 examples as unlabeled data. The baseline HMMs achieve F-scores of 50.8 and 76.3, respectively. In step 2, we add 7512 examples to each data set. After step 3, the new models achieve F-scores of 54.6 and 79.2, respectively. The gain for NER is lower than before as it is trained against "noisy" syntactic labels.

3.3 Analysis

Our goal is to prove that one-sided learning with hints "works." That is, if C_2 is learnable from large amounts of labeled data, then it is also learnable from small amounts of labeled data and large amounts of f_1-labeled data. This is formalized in Theorem 1 (all proofs are in Appendix A). However, before stating the theorem, we must define an "initial weakly-useful predictor" (terminology from Blum and Mitchell(1998)), and the notion of noisy PAC-learning in the structured domain.

Definition 2. *We say that h is a* weakly-useful predictor *of f if for all y:* $\Pr_{\mathcal{D}}[h(x) = y] \geq \epsilon$ *and* $\Pr_{\mathcal{D}}[f(x) = y \mid h(x) = y' \neq y] \geq \Pr_{\mathcal{D}}[f(x) = y] + \epsilon$.

This definition simply ensures that (1) h is non-trivial: it assigns some non-zero probability to every possible output; and (2) h is somewhat indicative of f. In practice, we use the hypothesis learned on the small amount of training data during step (1) of the algorithm as the weakly useful predictor.

Definition 3. *We say that $\mathcal{C}$ is* PAC-learnable with noise (in the structured setting) *if there exists an algorithm with the following properties. For any $c \in \mathcal{C}$, any distribution $\mathcal{D}$ over $\mathcal{X}$, any $0 \leq \eta \leq 1/|\mathcal{Y}|$, any $0 < \epsilon < 1$, any $0 < \delta < 1$ and any $\eta \leq \eta_0 < 1/|\mathcal{Y}|$, if the algorithm is given access to examples drawn $EX^{\eta}_{SN}(c, \mathcal{D})$ and inputs ϵ, δ and η_0, then with probability at least $1 - \delta$, the algorithm returns a hypothesis $h \in \mathcal{C}$ with error at most ϵ. Here, $EX^{\eta}_{SN}(c, \mathcal{D})$ is a structured noise oracle, which draws examples from $\mathcal{D}$, labels them by c and randomly replaces with another label with prob. η.*

Note here the rather weak notion of noise: entire structures are randomly changed, rather than individual labels. Furthermore, the error is 0/1 loss over the entire structure. Collins (2001) establishes learnability results for the class of hyperplane models under 0/1 loss. While not stated directly in terms of PAC learnability, it is clear that his results apply. Taskar et al. (2005) establish *tighter* bounds for the case of Hamming loss. This suggests that the requirement of 0/1 loss is weaker.

As suggested before, it is not sufficient for χ to simply be *correct* (the constant 1 function is correct, but not useful). We need it to be discriminating, made precise in the following definition.

Definition 4. *We say the* discrimination *of χ for h^0 is* $\Pr_{\mathcal{D}}[\chi(f_1(x), h^0(x))]^{-1}$.

In other words, a constraint function is discriminating when it is unlikely that our weakly-useful predictor h^0 chooses an output that satisfies the constraint. This means that if we *do* find examples (in our unlabeled corpus) that satisfy the constraints, they are likely to be "useful" to learning.

RUNNING EXAMPLE

In the NER HMM, let h^0 be the HMM obtained by training on the small labeled NER data set and f_1 is the true syntactic labels. We approximate $\Pr_{\mathcal{D}}$ by an empirical estimate over the unlabeled distribution. This gives a discrimination is 41.6 for the constraint function defined previously. However, if we compare against "weaker" constraint functions, we see the appropriate trend. The value for the constraint based only on POS tags is 39.1 (worse) and for the NP constraint alone is 27.0 (much worse).

Theorem 1. *Suppose C_2 is PAC-learnable with noise in the structured setting, h_2^0 is a weakly useful predictor of f_2, and χ is correct with respect to $\mathcal{D}, f_1, f_2, h_2^0$, and has discrimination $\geq 2(|\mathcal{Y}| - 1)$. Then C_2 is also PAC-learnable with one-sided hints.*

The way to interpret this theorem is that it tells us that if the initial h_2 we learn in step 1 of the one-sided algorithm is "good enough" (in the sense that it is weakly-useful), then we can use it as specified by the remainder of the one-sided algorithm to obtain an arbitrarily good h_2 (via iterating).

The dependence on $|\mathcal{Y}|$ is the discrimination bound for χ is unpleasant for structured problems. If we wish to find M unlabeled examples that satisfy the hints, we'll need a total of at least $2M(|\mathcal{Y}| - 1)$ total. This dependence can be improved as follows. Suppose that our structure is represented by a graph over vertices V, each of which can take a label from a set Y. Then, $|\mathcal{Y}| = \left|Y^V\right|$, and our result requires that χ be discriminating on an order exponential in V. Under the assumption that χ decomposes over the graph structure (true for our example) and that C_2 is PAC-learnable with per-vertex noise, then the discrimination requirement drops to $2\,|V|\,(|Y| - 1)$.

RUNNING EXAMPLE

In NER, $|Y| = 9$ and $|V| \approx 26$. This means that the values from the previous example look not quite so bad. In the 0/1 loss case, they are compared to 10^{25}; in the Hamming case, they are compared to only 416. The ability of the one-sided algorithm follows the same trends as the discrimination values. Recall the baseline performance is 50.8. With both constraints (and a discrimination value of 41.6), we obtain a score of 58.9. With just the POS constraint (discrimination of 39.1), we obtain a score of 58.1. With just the NP constraint (discrimination of 27.0, we obtain a score of 54.5.

The final question is how one-sided learning relates to two-sided learning. The following definition and easy corollary shows that they are related in the obvious manner, but depends on a notion of uncorrelation between h_1^0 and h_2^0.

Definition 5. *We say that h_1 and h_2 are* uncorrelated *if* $\Pr_{\mathcal{D}}\left[h_1(x) = y_1 \mid h_2(x) = y_2, x\right] = \Pr_{\mathcal{D}}\left[h_1(x) = y_1 \mid x\right]$.

Corollary 1. *Suppose C_1 and C_2 are both PAC-learnable in the structured setting, h_1^0 and h_2^0 are weakly useful predictors of f_1 and f_2, and χ is correct with respect to $\mathcal{D}, f_1, f_2, h_1^0$ and h_2^0, and has discrimination $\geq 4(|\mathcal{Y}| - 1)^2$ (for 0/1 loss) or $\geq 4\,|V|^2\,(|Y| - 1)^2$ (for Hamming loss), and that h_1^0 and h_2^0 are uncorrelated. Then C_1 and C_2 are also PAC-learnable with two-sided hints.*

Unfortunately, Corollary 1 depends *quadratically* on the discrimination term, unlike Theorem 1.

4 Experiments

In this section, we describe our experimental results. We have already discussed some of them in the context of the running example. In Section 4.1, we briefly describe the data sets we use. A full description of the HMM implementation and its results are in Section 4.2. Finally, in Section 4.3, we present results based on a competitive, discriminatively-learned sequence labeling algorithm. All results for NER and chunking are in terms of F-score; all results for POS tagging are accuracy.

4.1 Data Sets

Our results are based on syntactic data drawn from the Penn Treebank (Marcus et al., 1993), specifically the portion used by CoNLL 2000 shared task (Tjong Kim Sang and Buchholz, 2000). Our NER data is from two sources. The first source is the CoNLL 2003 shared task date (Tjong Kim Sang and De Meulder, 2003) and the second source is the 2004 NIST Automatic Content Extraction (Weischedel, 2004). The ACE data constitute six separate data sets from six domains: weblogs (wl), newswire (nw), broadcast conversations (bc), United Nations (un), direct telephone speech (dts) and broadcast news (bn). Of these, bc, dts and bn are all speech data sets. All the examples from the previous sections have been limited to the CoNLL data.

4.2 HMM Results

The experiments discussed in the preceding sections are based on a generative hidden Markov model for both the NER and syntactic chunking/POS tagging tasks. The HMMs constructed use first-order transitions and emissions. The emission vocabulary is pruned so that any word that appears ≤ 1 time in the training data is replaced by a unique `*unknown*` token. The transition and emission probabilities are smoothed with Dirichlet smoothing, $\alpha = 0.001$ (this was not-aggressively tuned by hand on one setting). The HMMs are implemented as finite state models in the Carmel toolkit (Graehl and Knight, 2002).

The various compatibility functions are also implemented as finite state models. We implement them as a transducer *from* POS/chunk labels *to* NER labels (though through the reverse operation, they can obviously be run in the opposite direction). The construction is with a single state with transitions:

- (NNP,?) maps to B-* and I-*
- (?,B-NP) maps to B-* and O
- (?,I-NP) maps to B-*, I-* and O
- Single exception: (NNP,x), where x is *not* an NP tag maps to anything (this is simply to avoid empty composition problems). This occurs in 100 of the $212k$ words in the Treebank data and more rarely in the automatically tagged data.

4.3 One-sided Discriminative Learning

In this section, we describe the results of one-sided discriminative labeling with hints. We use the true syntactic labels from the Penn Treebank to derive the constraints (this is roughly 9000 sentences). We use the LaSO sequence labeling software (Daumé III and Marcu, 2005), with its built-in feature set.

Our goal is to analyze two things: (1) what is the effect of the amount of labeled NER data? (2) what is the effect of the amount of labeled syntactic data from which the hints are constructed?

To answer the first question, we keep the amount of syntactic data fixed (at 8936 sentences) and vary the amount of NER data in $N \in \{100, 200, 400, 800, 1600\}$. We compare models with and without the default gazetteer information from the LaSO software. We have the following models for comparison:

- A default "Baseline" in which we simply train the NER model without using syntax.

	Hints vs **Base**	**Self-T** vs **Base**	**Hints** vs **Self-T**
Win	29	20	24
Tie	6	12	11
Lose	0	3	0

Table 1: Comparison between hints, self-training and the (best) baseline for varying amount of labeled data.

- In "POS-feature", we do the same thing, but we first label the NER data using a tagger/chunker trained on the 8936 syntactic sentences. These labels are used as features for the baseline.
- A "Self-training" setting where we use the 8936 syntactic sentences as "unlabeled," label them with our model, and then train on the results. (This is equivalent to a hints model where $\chi(\cdot,\cdot) = 1$ is the constant 1 function.) We use model confidence as in Blum and Mitchell (1998).[4]

The results are shown in Figure 1. The trends we see are the following:

- More data always helps.
- Self-training usually helps over the baseline (though not always: for instance in wl and parts of cts and bn).
- Adding the gazetteers help.
- Adding the syntactic features helps.
- Learning with hints, especially for ≤ 1000 training data points, helps significantly, even over self-training.

We further compare the algorithms by looking at how many training setting has each as the winner. In particular, we compare both hints and self-training to the two baselines, and then compare hints to self-training. If results are not significant at the 95% level (according to McNemar's test), we call it a tie. The results are in Table 1.

In our second set of experiments, we consider the role of the syntactic data. For this experiment, we hold the number of NER labeled sentences constant (at $N = 200$) and vary the amount of syntactic data in $M \in \{500, 1000, 2000, 4000, 8936\}$. The results of these experiments are in Figure 2. The trends are:

- The POS feature is relatively insensitive to the amount of syntactic data—this is most likely because it's weight is discriminatively adjusted

[4]Results without confidence were significantly worse.

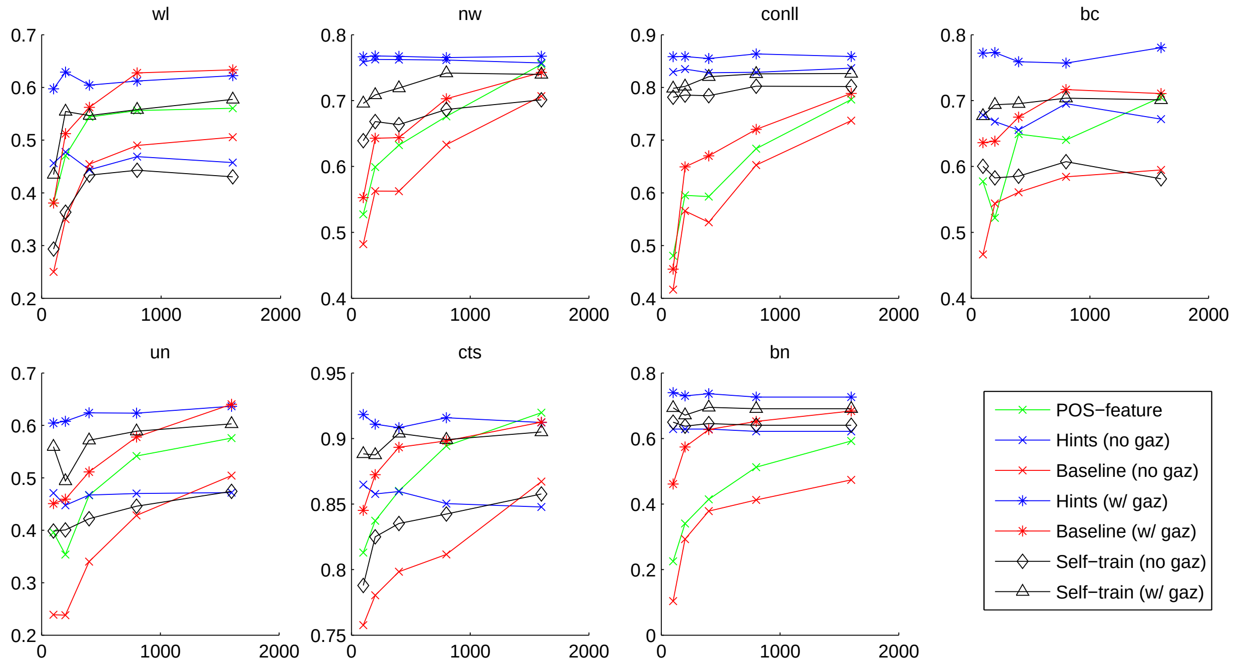

Figure 1: Results of varying the amount of NER labeled data, for a fixed amount ($M = 8936$) of syntactic data.

	Hints vs **Base**	**Self-T** vs **Base**	**Hints** vs **Self-T**
Win	34	28	15
Tie	1	7	20
Lose	0	0	0

Table 2: Comparison between hints, self-training and the (best) baseline for varying amount of unlabeled data.

by LaSO so that if the syntactic information is bad, it is relatively ignored.

- Self-training performance often *degrades* as the amount of syntactic data increases.
- The performance of learning with hints increases steadily with more syntactic data.

As before, we compare performance between the different models, declaring a "tie" if the difference is not statistically significant at the 95% level. The results are in Table 2.

In experiments not reported here to save space, we experimented with several additional settings. In one, we weight the unlabeled data in various ways: (1) to make it equal-weight to the labeled data; (2) at 10% weight; (3) according to the score produced by the first round of labeling. None of these had a significant impact on scores; in a few cases performance went up by $\ll 1$, in a few cases, performance went down about the same amount.

4.4 Two-sided Discriminative Learning

In this section, we explore the use of two-sided discriminative learning to boost the performance of our syntactic chunking, part of speech tagging, and named-entity recognition software. We continue to use LaSO (Daumé III and Marcu, 2005) as the sequence labeling technique.

The results we present are based on attempting to improve the performance of a state-of-the-art system train on *all* of the training data. (This is in contrast to the results in Section 4.3, in which the effect of using limited amounts of data was explored.) For the POS tagging and syntactic chunking, we being with all 8936 sentences of training data from CoNLL. For the named entity recognition, we limit our presentation to results from the CoNLL 2003 NER shared task. For this data, we have roughly $14k$ sentences of training data, all of which are used. In both cases, we reserve 10% as development data. The development data is use to do early stopping in LaSO.

As unlabeled data, we use $1m$ sentences extracted from the North American National Corpus of En-

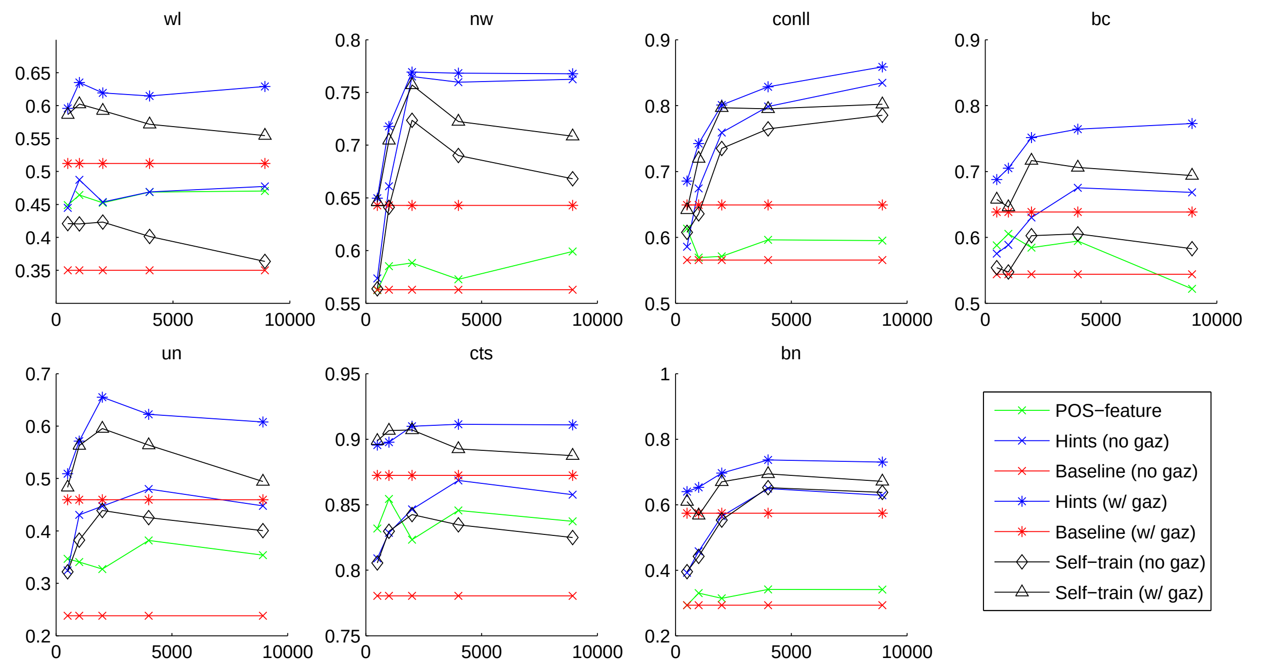

Figure 2: Results of varying amount of syntactic data for a fixed amount of NER data ($N = 200$ sentences).

glish (previously used for self-training of parsers (McClosky et al., 2006)). These $1m$ sentences were selected by dev-set relativization against the union of the two development data sets.

Following similar ideas to those presented by Blum and Mitchell (1998), we employ two slight modifications to the algorithm presented in Section 3.2. First, in step (2b) instead of adding *all* allowable instances to the labeled data set, we only add the top R (for some hyper-parameter R), where "top" is determined by average model confidence for the two tasks. Second, Instead of using the *full* unlabeled set to label at each iteration, we begin with a random subset of $10R$ unlabeled examples and another add random $10R$ every iteration.

We use the same baseline systems as in one-sided learning: a Baseline that learns the two tasks independently; a variant of the Baseline on which the output of the POS/chunker is used as a feature for the NER; a variant based on self-training; the hints-based method. In all cases, we *do* use gazetteers. We run the hints-based model for 10 iterations. For self-training, we use $10R$ unlabeled examples (so that it had access to the same amount of unlabeled data as the hints-based learning after all 10 iterations). We used three values of R: 50, 100, 500. We select the

	Chunking	**NER**
Baseline	94.2	87.5
w/POS	N/A	88.0
Self-train		
$R = 50$	94.2	88.0
$R = 100$	94.3	88.6
$R = 500$	94.1	88.4
Hints		
$R = 50$	94.2	88.5
$R = 100$	94.3	89.1
$R = 500$	94.3	89.0

Table 3: Results on two-sided learning with hints.

best-performing model (by the dev data) over these ten iterations. The results are in Table 3.

As we can see, performance for syntactic chunking is relatively stagnant: there are no significant improvements for any of the methods over the baseline. This is not surprising: the form of the constraint function we use tells us a *lot* about the NER task, but relatively little about the syntactic chunking task. In particular, it tells us nothing about phrases other than NPs. On the other hand, for NER, we see that both self-training and learning with hints improve over the baseline. The improvements are not enormous, but *are* significant (at the 95% level, as measured by McNemar's test). Unfortunately, the improvements for learning with hints over the self-training model are only significant at the 90% level.

5 Discussion

We have presented a method for simultaneously learning two tasks using prior knowledge about the relationship between their outputs. This is related to *joint inference* (Daumé III et al., 2006). However, we do not require that that a single data set be labeled for multiple tasks. In all our examples, we use separate data sets for shallow parsing as for named-entity recognition. Although all our experiments used the LaSO framework for sequence labeling, there is *noting* in our method that assumes any particular learner; alternatives include: conditional random fields (Lafferty et al., 2001), independent predictors (Punyakanok and Roth, 2001), max-margin Markov networks (Taskar et al., 2005), etc.

Our approach, both algorithmically and theoretically, is most related to ideas in co-training (Blum and Mitchell, 1998). The key difference is that in co-training, one assumes that the two "views" are on the *inputs*; here, we can think of the two output spaces as being the difference "views" and the compatibility function χ being a method for reconciling these two views. Like the pioneering work of Blum and Mitchell, the algorithm we employ in practice makes use of incrementally augmenting the unlabeled data and using model confidence. Also like that work, we do not currently have a theoretical framework for this (more complex) model.[5] It would also be interesting to explore *soft* hints, where the range of χ is $[0, 1]$ rather than $\{0, 1\}$.

Recently, Ganchev et al. (2008) proposed a co-regularization framework for learning across multiple related tasks with different output spaces. Their approach hinges on a constrained EM framework and addresses a quite similar problem to that addressed by this paper. Chang et al. (2008) also propose a "semisupervised" learning approach quite similar to our own model. The show very promising results in the context of semantic role labeling. Given the apparent (very!) recent interest in this problem, it would be ideal to directly compare the different approaches.

In addition to an analysis of the theoretical properties of the algorithm presented, the most compelling avenue for future work is to apply this framework to other task pairs. With a little thought, one can imagine formulating compatibility functions between tasks like discourse parsing and summarization (Marcu, 2000), parsing and word alignment, or summarization and information extraction.

Acknowledgments

Many thanks to three anonymous reviewers of this papers whose suggestions greatly improved the work and the presentation. This work was partially funded by NSF grant IIS 0712764.

A Proofs

The proof of Theorem 1 closes follows that of Blum and Mitchell (1998).

Proof (Theorem 1, sketch). Use the following notation: $c_k = \Pr_{\mathcal{D}}[h(x) = k]$, $p_l = \Pr_{\mathcal{D}}[f(x) = l]$, $q_{l|k} = \Pr_{\mathcal{D}}[f(x) = l \mid h(x) = k]$. Denote by $\mathcal{A}$ the set of outputs that satisfy the constraints. We are interested in the probability that $h(x)$ is erroneous, given that $h(x)$ satisfies the constraints:

$$\begin{aligned} & p\left(h(x) \in \mathcal{A}\backslash\{l\} \mid f(x) = l\right) \\ & = \sum_{k \in \mathcal{A}\backslash\{l\}} p\left(h(x) = k \mid f(x) = l\right) = \sum_{k \in \mathcal{A}\backslash\{l\}} c_k q_{l|k}/p_l \\ & \leq \sum_{k \in \mathcal{A}} c_k(|\mathcal{Y}| - 1 + \epsilon \sum_{l \neq k} 1/p_l) \leq 2 \sum_{k \in \mathcal{A}} c_k(|\mathcal{Y}| - 1) \end{aligned}$$

Here, the second step is Bayes' rule plus definitions, the third step is by the weak initial hypothesis assumption, and the last step is by algebra. Thus, in order to get a probability of error at most η, we need $\sum_{k \in \mathcal{A}} c_k = \Pr[h(x) \in \mathcal{A}] \leq \eta/(2(|\mathcal{Y}| - 1))$. $\square$

The proof of Corollary 1 is straightforward.

Proof (Corollary 1, sketch). Write out the probability of error as a double sum over true labels y_1, y_2 and predicted labels $\hat{y}_1, \hat{y}_2$ subject to $\chi(\hat{y}_1, \hat{y}_2)$. Use the uncorrelation assumption and Bayes' to split this into the product two terms as in the proof of Theorem 1. Bound as before. $\square$

[5] Dasgupta et al. (2001) proved, three years later, that a formal model roughly equivalent to the actual Blum and Mitchell algorithm *does* have solid theoretical foundations.

References

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, pages 92–100.

Ming-Wei Chang, Lev Ratinov, Nicholas Rizzolo, and Dan Roth. 2008. Learning and inference with constraints. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Michael Collins. 2001. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In *International Workshop on Parsing Technologies (IWPT)*.

Sanjoy Dasgupta, Michael Littman, and David McAllester. 2001. PAC generalization bounds for co-training. In *Advances in Neural Information Processing Systems (NIPS)*.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Hal Daumé III, Andrew McCallum, Ryan McDonald, Fernando Pereira, and Charles Sutton, editors. 2006. *Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technology (NAACL/HLT).

Kuzman Ganchev, Joao Graca, John Blitzer, and Ben Taskar. 2008. Multi-view learning over structured and non-identical outputs. In *Proceedings of the Converence on Uncertainty in Artificial Intelligence (UAI)*.

Jonathan Graehl and Kevin Knight. 2002. Carmel finite state transducer package. `http://www.isi.edu/licensed-sw/carmel/`.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. The MIT Press, Cambridge, Massachusetts.

Mitch Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technology (NAACL/HLT)*.

Vasin Punyakanok and Dan Roth. 2001. The use of classifiers in sequential inference. In *Advances in Neural Information Processing Systems (NIPS)*.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 897–904.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of Conference on Computational Natural Language Learning*, pages 142–147.

Nicola Ueffing. 2006. Self-training for machine translation. In *NIPS workshop on Machine Learning for Multilingual Information Access*.

Leslie G. Valiant. 1994. A theory of the learnable. *Annual ACM Symposium on Theory of Computing*, pages 436–445.

Ralph Weischedel, editor. 2004. *Automatic Content Extraction Workshop (ACE-2004)*, Alexandria, Virginia, September 20–22.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.