Bootstrapping via Graph Propagation

Max Whitney and **Anoop Sarkar**[*]
Simon Fraser University, School of Computing Science
Burnaby, BC V5A 1S6, Canada
{mwhitney,anoop}@sfu.ca

Abstract

Bootstrapping a classifier from a small set of seed rules can be viewed as the propagation of labels between examples via features shared between them. This paper introduces a novel variant of the Yarowsky algorithm based on this view. It is a bootstrapping learning method which uses a graph propagation algorithm with a well defined objective function. The experimental results show that our proposed bootstrapping algorithm achieves state of the art performance or better on several different natural language data sets.

1 Introduction

In this paper, we are concerned with a case of semi-supervised learning that is close to unsupervised learning, in that the labelled and unlabelled data points are from the same domain and only a small set of seed rules is used to derive the labelled points. We refer to this setting as *bootstrapping*. In contrast, typical semi-supervised learning deals with a large number of labelled points, and a domain adaptation task with unlabelled points from the new domain.

The two dominant discriminative learning methods for bootstrapping are self-training (Scudder, 1965) and co-training (Blum and Mitchell, 1998). In this paper we focus on a self-training style bootstrapping algorithm, the Yarowsky algorithm (Yarowsky, 1995). Variants of this algorithm have been formalized as optimizing an objective function in previous work by Abney (2004) and Haffari and Sarkar (2007), but it is not clear that any perform as well as the Yarowsky algorithm itself.

We take advantage of this formalization and introduce a novel algorithm called Yarowsky-prop which builds on the algorithms of Yarowsky (1995) and Subramanya et al. (2010). It is theoretically well-understood as minimizing an objective function at each iteration, and it obtains state of the art performance on several different NLP data sets. To our knowledge, this is the first theoretically motivated self-training bootstrapping algorithm which performs as well as the Yarowsky algorithm.

[*]This research was partially supported by an NSERC, Canada (RGPIN: 264905) grant. We would like to thank Gholamreza Haffari and the anonymous reviewers for their comments. We particularly thank Michael Collins, Jason Eisner, and Damianos Karakos for the data we used in our experiments.

x	denotes an example
f, g	denote features
i, k	denote labels
X	set of training examples
F_x	set of features for example x
Y	current labelling of X
Y_x	current label for example x
$\perp$	value of Y_x for unlabelled examples
L	number of labels (not including $\perp$)
Λ	set of currently labelled examples
V	set of currently unlabelled examples
X_f	set of examples with feature f
Λ_f	set of currently labelled examples with f
V_f	set of currently unlabelled examples with f
Λ_j	set of examples currently labelled with j
Λ_{fj}	set of examples with f currently labelled with j

Table 1: Notation of Abney (2004).

2 Bootstrapping

Abney (2004) defines useful notation for semi-supervised learning, shown in table 1. Note that Λ, V, etc. are relative to the current labelling Y. We additionally define F to be the set of all features, and use U to denote the uniform distribution. In the bootstrapping setting the learner is given an initial partial labelling $Y^{(0)}$ where only a few examples are labelled (i.e. $Y_x^{(0)} = \perp$ for most x).

Abney (2004) defines three probability distributions in his analysis of bootstrapping: θ_{fj} is the parameter for feature f with label j, taken to be normalized so that θ_f is a distribution over labels. ϕ_x is the labelling distribution representing the current Y; it is a point distribution for labelled examples and uniform for unlabelled examples. π_x is the prediction distribution over labels for example x.

The approach of Haghighi and Klein (2006b) and Haghighi and Klein (2006a) also uses a small set of

Algorithm 1: The basic Yarowsky algorithm.

```
Require: training data $X$ and a seed DL $\theta^{(0)}$
1: apply $\theta^{(0)}$ to $X$ produce a labelling $Y^{(0)}$
2: for iteration $t$ to maximum or convergence do
3:     train a new DL $\theta$ on $Y^{(t)}$
4:     apply $\theta$ to $X$, to produce $Y^{(t+1)}$
5: end for
```

seed rules but uses them to inject features into a joint model $p(x, j)$ which they train using expectation-maximization for Markov random fields. We focus on discriminative training which does not require complex partition functions for normalization. Blum and Chawla (2001) introduce an early use of transductive learning using graph propagation. X. Zhu and Z. Ghahramani and J. Lafferty (2003)'s method of graph propagation is predominantly transductive, and the non-transductive version is closely related to Abney (2004) c.f. Haffari and Sarkar (2007).

3 Existing algorithms

3.1 Yarowsky

A decision list (DL) is a (ordered) list of feature-label pairs (rules) which is produced by assigning a score to each rule and sorting on this score. It chooses a label for an example from the first rule whose feature is a feature of the example. For a DL the prediction distribution is defined by $\pi_x(j) \propto \max_{f \in F_x} \theta_{fj}$. The basic Yarowsky algorithm is shown in algorithm 1. Note that at any point some training examples may be left unlabelled by $Y^{(t)}$.

We use Collins and Singer (1999) for our exact specification of Yarowsky.[1] It uses DL rule scores

$$\theta_{fj} \propto \frac{|\Lambda_{fj}| + \epsilon}{|\Lambda_f| + L\epsilon} \tag{1}$$

where ϵ is a smoothing constant. When constructing a DL it keeps only the rules with (pre-normalized) score over a threshold ζ. In our implementation we add the seed rules to each subsequent DL.[2]

3.2 Yarowsky-cautious

Collins and Singer (1999) also introduce a variant algorithm Yarowsky-cautious. Here the DL training step keeps only the top n rules (f, j) over the threshold for each label j, ordered by $|\Lambda_f|$. Additionally the threshold ζ is checked against $|\Lambda_{fj}|/|\Lambda_f|$ instead of the smoothed score. n begins at n_0 and is incremented by Δn at each iteration. We add the seed DL to the new DL after applying the cautious pruning. Cautiousness limits not only the size of the DL but also the number of labelled examples, prioritizing decisions which are believed to be of high accuracy.

At the final iteration Yarowsky-cautious uses the current labelling to train a DL without a threshold or cautiousness, and this DL is used for testing. We call this the retraining step.[3]

3.3 DL-CoTrain

Collins and Singer (1999) also introduce the co-training algorithm DL-CoTrain. This algorithm alternates between two DLs using disjoint views of the features in the data. At each step it trains a DL and then produces a new labelling for the other DL. Each DL uses thresholding and cautiousness as we describe for Yarowsky-cautious. At the end the DLs are combined, the result is used to label the data, and a retraining step is done from this single labelling.

3.4 Y-1/DL-1-VS

One of the variant algorithms of Abney (2004) is Y-1/DL-1-VS (referred to by Haffari and Sarkar (2007) as simply DL-1). Besides various changes in the specifics of how the labelling is produced, this algorithm has two differences versus Yarowsky. Firstly, the smoothing constant ϵ in (1) is replaced by $1/|V_f|$. Secondly, π is redefined as $\pi_x(j) = \frac{1}{|F_x|} \sum_{f \in F_x} \theta_{fj}$, which we refer to as the sum definition of π. This definition does not match a literal DL but is easier to analyze.

We are not concerned here with the details of Y-1/DL-1-VS, but we note that Haffari and Sarkar (2007) provide an objective function for this algorithm using a generalized definition of cross-entropy in terms of Bregman distance, which motivates our objective in section 4. The Breg-

[1] It is similar to that of Yarowsky (1995) but is better specified and omits word sense disambiguation optimizations. The general algorithm in Yarowsky (1995) is self-training with any kind of underlying supervised classifier, but we follow the convention of using Yarowsky to refer to the DL algorithm.

[2] This is not clearly specified in Collins and Singer (1999), but is used for DL-CoTrain in the same paper.

[3] The details of Yarowsky-cautious are not clearly specified in Collins and Singer (1999). Based on similar parts of DL-CoTrain we assume the that the top n selection is per label rather in total, that the thresholding value is unsmoothed, and that there is a retraining step. We also assume their notation *Count*$'(x)$ to be equivalent to $|\Lambda_f|$.

man distance between two discrete probability distributions p and q is defined as $B_\psi(p,q) = \sum_i [\psi(p_i) - \psi(q_i) - \psi'(q_i)(p_i - q_i)]$. As a specific case we have $B_{t^2}(p,q) = \sum_i (p_i - q_i)^2 = ||p-q||^2$. Then Bregman distance-based entropy is $H_{t^2}(p) = -\sum_i p_i^2$, KL-Divergence is B_{t^2}, and cross-entropy follows the standard definition in terms of H_{t^2} and B_{t^2}. The objective minimized by Y-1/DL-1-VS is:

$$\sum_{\substack{x \in X \\ f \in F_x}} H_{t^2}(\phi_x || \theta_f) = \sum_{\substack{x \in X \\ f \in F_x}} \left[B_{t^2}(\phi_x || \theta_f) - \sum_y \phi_x^2 \right]. \tag{2}$$

3.5 Yarowsky-sum

As a baseline for the sum definition of π, we introduce the Yarowsky-sum algorithm. It is the same as Yarowsky except that we use the sum definition when labelling: for example x we choose the label j with the highest (sum) $\pi_x(j)$, but set $Y_x = \perp$ if the sum is zero. Note that this is a linear model similar to a conditional random field (CRF) (Lafferty et al., 2001) for unstructured multiclass problems.

3.6 Bipartite graph algorithms

Haffari and Sarkar (2007) suggest a bipartite graph framework for semi-supervised learning based on their analysis of Y-1/DL-1-VS and objective (2). The graph has vertices $X \cup F$ and edges $\{(x,f) : x \in X, f \in F_x\}$, as in the graph shown in figure 1(a). Each vertex represents a distribution over labels, and in this view Yarowsky can be seen as alternately updating the example distributions based on the feature distributions and visa versa.

Based on this they give algorithm 2, which we call HS-bipartite. It is parametrized by two functions which are called features-to-example and examples-to-feature here. Each can be one of two choices: **average**(S) is the normalized average of the distributions of S, while **majority**(S) is a uniform distribution if all labels are supported by equal numbers of distributions of S, and otherwise a point distribution with mass on the best supported label. The **average**-**majority** form is similar to Y-1/DL-1-VS, and the **majority**-**majority** form minimizes a different objective similar to (2).

In our implementation we label training data (for the convergence check) with the ϕ distributions from the graph. We label test data by constructing new $\phi_x =$ examples-to-feature(F_x) for the unseen x.

Algorithm 2: HS-bipartite.

```
1: apply $\theta^{(0)}$ to $X$ produce a labelling $Y^{(0)}$
2: **for** iteration $t$ to maximum or convergence **do**
3:     **for** $f \in F$ **do**
4:         let $p =$ examples-to-feature$(\{\phi_x : x \in X_f\})$
5:         if $p \neq U$ then let $\theta_f = p$
6:     **end for**
7:     **for** $x \in X$ **do**
8:         let $p =$ features-to-example$(\{\theta_f : f \in F_x\})$
9:         if $p \neq U$ then let $\phi_x = p$
10:    **end for**
11: **end for**
```

3.7 Semi-supervised learning algorithm of Subramanya et al. (2010)

Subramanya et al. (2010) give a semi-supervised algorithm for part of speech tagging. Unlike the algorithms described above, it is for domain adaptation with large amounts of labelled data rather than bootstrapping with a small number of seeds.

This algorithm is structurally similar to Yarowsky in that it begins from an initial partial labelling and repeatedly trains a classifier on the labelling and then relabels the data. It uses a CRF (Lafferty et al., 2001) as the underlying supervised learner. It differs significantly from Yarowsky in two other ways: First, instead of only training a CRF it also uses a step of graph propagation between distributions over the n-grams in the data. Second, it does the propagation on distributions over n-gram types rather than over n-gram tokens (instances in the data).

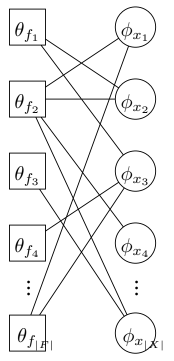
They argue that using propagation over types allows the algorithm to enforce constraints and find similarities that self-training cannot. We are not concerned here with the details of this algorithm, but it motivates our work firstly in providing the graph propagation which we will describe in more detail in section 4, and secondly in providing an algorithmic structure that we use for our algorithm in section 5.

3.8 Collins and Singer (1999)'s EM

We implemented the EM algorithm of Collins and Singer (1999) as a baseline for the other algorithms. They do not specify tuning details, but to get comparable accuracy we found it was necessary to do smoothing and to include weights λ_1 and λ_2 on the expected counts of seed-labelled and initially unla-

Method	$\mathcal{V}$	$\mathcal{N}(u)$	q_u
ϕ-θ	$X \cup F$	$\mathcal{N}_x = F_x, \mathcal{N}_f = X_f$	$q_x = \phi_x, q_f = \theta_f$
π-θ	$X \cup F$	$\mathcal{N}_x = F_x, \mathcal{N}_f = X_f$	$q_x = \pi_x, q_f = \theta_f$
θ-only	F	$\mathcal{N}_f = \bigcup_{x \in X_f} F_x \setminus f$	$q_f = \theta_f$
θ^{T}-only	F	$\mathcal{N}_f = \bigcup_{x \in X_f} F_x \setminus f$	$q_f = \theta^{\mathrm{T}}_f$

Table 2: Graph structures for propagation.


(a) ϕ-θ method

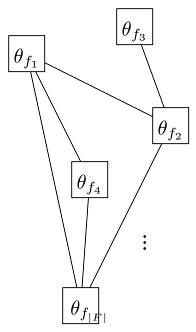

(b) θ-only method

Figure 1: Example graphs for ϕ-θ and θ-only propagation.

belled examples respectively (Nigam et al., 2000).

4 Graph propagation

The graph propagation of Subramanya et al. (2010) is a method for smoothing distributions attached to vertices of a graph. Here we present it with an alternate notation using Bregman distances as described in section 3.4.[4] The objective is

$$\mu \sum_{\substack{u \in \mathcal{V} \\ v \in \mathcal{N}(i)}} w_{uv} B_{t^2}(q_u, q_v) + \nu \sum_{u \in V} B_{t^2}(q_u, U) \quad (3)$$

where $\mathcal{V}$ is a set of vertices, $\mathcal{N}(v)$ is the neighbourhood of vertex v, and q_v is an initial distribution for each vertex v to be smoothed. They give an iterative update to minimize (3). Note that (3) is independent of their specific graph structure, distributions, and semi-supervised learning algorithm.

We propose four methods for using this propagation with Yarowsky. These methods all use constant edge weights ($w_{uv} = 1$). The distributions and graph structures are shown in table 2. Figure 1 shows example graphs for ϕ-θ and θ-only. π-θ and θ^{T}-only are similar, and are described below.

The graph structure of ϕ-θ is the bipartite graph of Haffari and Sarkar (2007). In fact, ϕ-θ the propagation objective (3) and Haffari and Sarkar (2007)'s Y-1/DL-1-VS objective (2) are identical up to constant coefficients and an extra constant term.[5] ϕ-θ therefore gives us a direct way to optimize (2).

The other three methods do not correspond to the objective of Haffari and Sarkar (2007). The π-θ method is like ϕ-θ except for using π as the distribution for example vertices.

The bipartite graph of the first two methods differs from the structure used by Subramanya et al. (2010) in that it does propagation between two different kinds of distributions instead of only one kind. We also adopt a more comparable approach with a graph over only features. Here we define adjacency by co-occurrence in the same example. The θ-only method uses this graph and θ as the distribution.

Finally, we noted in section 3.7 that the algorithm of Subramanya et al. (2010) does one additional step in converting from token level distributions to type level distributions. The θ^{T}-only method therefore uses the feature-only graph but for the distribution uses a type level version of θ defined by $\theta^{\mathrm{T}}_{fj} = \frac{1}{|X_f|} \sum_{x \in X_f} \pi_x(j)$.

5 Novel Yarowsky-prop algorithm

We call our graph propagation based algorithm Yarowsky-prop. It is shown with θ^{T}-only propagation in algorithm 3. It is based on the Yarowsky algorithm, with the following changes: an added step to calculate θ^{T} (line 4), an added step to calculate θ^{P} (line 5), the use of θ^{P} rather than the DL to update the labelling (line 6), and the use of the sum definition of π. Line 7 does DL training as we describe in sections 3.1 and 3.2. Propagation is done with the iterative update of Subramanya et al. (2010).

This algorithm is adapted to the other propagation methods described in section 4 by changing the type of propagation on line 5. In θ-only, propagation is done on θ, using the graph of figure 1(b). In ϕ-θ and π-θ propagation is done on the respective bipartite graph (figure 1(a) or the equivalent with π). Line

[4] We omit the option to hold some of the distributions at fixed values, which would add an extra term to the objective.

[5] The differences are specifically: First, (3) adds the constant coefficients μ and ν. Second, (3) sums over each edge twice (once in each direction), while (2) sums over each only once. Since $w_{uv} = w_{vu}$ and $B_{t^2}(q_u, q_v) = B_{t^2}(q_v, q_u)$, this can be folded into the constant μ. Third, after expanding (2) there is a term $|F_x|$ inside the sum for $H_{t^2}(\phi_x)$ which is not present in (3). This does not effect the direction of minimization. Fourth, $B_{t^2}(q_u, U)$ in (3) expands to $H_{t^2}(q_u)$ plus a constant, adding an extra constant term to the total.

Algorithm 3: Yarowsky-prop.

```
1: let $\theta_{fj}$ be the scores of the seed rules // crf_train
2: for iteration $t$ to maximum or convergence do
3:     let $\pi_x(j) = \frac{1}{|F_x|} \sum_{f \in F_x} \theta_{fj}$ // post._decode
4:     let $\theta^{\mathrm{T}}_{fj} = \frac{\sum_{x \in X_f} \pi_x(j)}{|X_f|}$ // token_to_type
5:     propagate $\theta^{\mathrm{T}}$ to get $\theta^{\mathrm{P}}$ // graph_propagate
6:     label the data with $\theta^{\mathrm{P}}$ // viterbi_decode
7:     train a new DL $\theta_{fj}$ // crf_train
8: end for
```

4 is skipped for these methods, and ϕ is as defined in section 2. For the bipartite graph methods ϕ-θ and π-θ only the propagated θ values on the feature nodes are used for θ^P (the distributions on the example nodes are ignored after the propagation itself).

The algorithm uses θ_{fj} values rather than an explicit DL for labelling. The (pre-normalized) score for any (f, j) not in the DL is taken to be zero. Besides using the sum definition of π when calculating θ^{T}, we also use a sum in labelling. When labelling an example x (at line 6 and also on testing data) we use $\arg\max_j \sum_{f \in F_x:\, \theta^{\mathrm{P}}_f \neq U} \theta^{\mathrm{P}}_{fj}$, but set $Y_x = \perp$ if the sum is zero. Ignoring uniform θ^{P}_f values is intended to provide an equivalent to the DL behaviour of using evidence only from rules that are in the list.

We include the cautiousness of Yarowsky-cautious (section 3.2) in the DL training on line 7. At the labelling step on line 6 we label only examples which the pre-propagated θ would also assign a label (using the same rules described above for θ^{P}). This choice is intended to provide an equivalent to the Yarowsky-cautious behaviour of limiting the number of labelled examples; most θ^{P}_f are non-uniform, so without it most examples become labelled early.

We observe further similarity between the Yarowsky algorithm and the general approach of Subramanya et al. (2010) by comparing algorithm 3 here with their algorithm 1. The comments in algorithm 3 give the corresponding parts of their algorithm. Note that each line has a similar purpose.

6 Evaluation

6.1 Tasks and data

For evaluation we use the tasks of Collins and Singer (1999) and Eisner and Karakos (2005), with data kindly provided by the respective authors.

The task of Collins and Singer (1999) is named entity classification on data from New York Times text.[6] The data set was pre-processed by a statistical parser (Collins, 1997) and all noun phrases that are potential named entities were extracted from the parse tree. Each noun phrase is to be labelled as a person, organization, or location. The parse tree provides the surrounding context as *context features* such as the words in prepositional phrase and relative clause modifiers, etc., and the actual words in the noun phrase provide the *spelling features*. The test data additionally contains some noise examples which are not in the three named entity categories. We use the seven seed rules which the authors provide: “New-York”, “California”, and “U.S.” indicate locations, “Microsoft”, “I.B.M.”, and “Incorporated” indicate organizations, and ‘Mr.” in a word indicates a person. For DL-CoTrain, we use their two views: one view is the spelling features, and the other is the context features. Figure 2 shows a DL from Yarowsky training on this task.

Rank	Score	Feature	Label
1	0.999900	New-York	loc.
2	0.999900	California	loc.
3	0.999900	U.S.	loc.
4	0.999900	Microsoft	org.
5	0.999900	I.B.M.	org.
6	0.999900	Incorporated	org.
7	0.999900	Mr.	per.
8	0.999976	U.S.	loc.
9	0.999957	New-York-Stock-Exchange	loc.
10	0.999952	California	loc.
11	0.999947	New-York	loc.
12	0.999946	*court-in*	loc.
13	0.975154	*Company-of*	loc.
		⋮	

Figure 2: A DL from iteration 5 of Yarowsky on the named entity task. Scores are pre-normalized values from the expression on the left side of (1), not θ_{fj} values. Context features are indicated by *italics*; all others are spelling features. Specific feature types are omitted. Seed rules are indicated by **bold** ranks.

The tasks of Eisner and Karakos (2005) are word sense disambiguation on several English words which have two senses corresponding to two different words in French. Data was extracted from the Canadian Hansards, using the English side to produce training and test data and the French side to produce the gold labelling. Features are the original and lemmatized words immediately adja-

[6] We removed weekday and month examples from the test set as they describe. They note 88962 examples in their training set, but the file has 89305. We did not find any filtering criteria that produced the expected size, and therefore used all examples.

cent to the word to be disambiguated, and original and lemmatized context words in the same sentence. Their seeds are pairs of adjacent word features, with one feature for each label (sense). We use the 'drug', 'land', and 'sentence' tasks, and the seed rules from their best seed selection: 'alcohol'/'medical', 'acres'/'court', and 'reads'/'served' respectively (they do not provide seeds for their other three tasks). For DL-CoTrain we use adjacent words for one view and context words for the other.

6.2 Experimental set up

Where applicable we use smoothing $\epsilon = 0.1$, a threshold $\zeta = 0.95$, and cautiousness parameters $n_0 = \Delta n = 5$ as in Collins and Singer (1999) and propagation parameters $\mu = 0.6, \nu = 0.01$ as in Subramanya et al. (2010). Initial experiments with different propagation parameters suggested that as long as ν was set at this value changing μ had relatively little effect on the accuracy. We did not find any propagation parameter settings that outperformed this choice. For the Yarowsky-prop algorithms we perform a single iteration of the propagation update for each iteration of the algorithm.

For EM we use weights $\lambda_1 = 0.98$, and $\lambda_2 = 0.02$ (see section 3.8), which were found in initial experiments to be the best values, and results are averaged over 10 random initializations.

The named entity test set contains some examples that are neither person, organization, nor location. Collins and Singer (1999) define noise accuracy as accuracy that includes such instances, and clean accuracy as accuracy calculated across only the examples which are one of the known labels. We report only clean accuracy in this paper; noise accuracy tracks clean accuracy but is a little lower. There is no difference on the word sense data sets. We also report (clean) non-seeded accuracy, which we define to be clean accuracy over only examples which are not assigned a label by the seed rules. This is intended to evaluate what the algorithm has learned, rather than what it can achieve by using the input information directly (Daume, 2011).

We test Yarowsky, Yarowsky-cautious, Yarowsky-sum, DL-CoTrain, HS-bipartite in all four forms, and Yarowsky-prop cautious and non-cautious and in all four forms. For each algorithm except EM we perform a final retraining step as described for Yarowsky-cautious (section 3.2). Our programs and experiment scripts have been made available.[7]

Gold	Spelling features	Context features
loc.	Waukegan	*maker*, *LEFT*
loc.	Mexico, president, of	*president-of*, *RIGHT*
loc.	La-Jolla, La Jolla	*company*, *LEFT*

Figure 3: Named entity test set examples where Yarowsky-prop θ-only is correct and no other tested algorithms are correct. The specific feature types are omitted.

6.3 Accuracy

Table 3 shows the final test set accuracies for the all the algorithms. The seed DL accuracy is also included for reference.

The best performing form of our novel algorithm is Yarowsky-prop-cautious θ-only. It numerically outperforms DL-CoTrain on the named entity task, is not (statistically) significantly worse on the drug and land tasks, and is significantly better on the sentence task. It also numerically outperforms Yarowsky-cautious on the named entity task and is significantly better on the drug task. Is significantly worse on the land task, where most algorithms converge at labelling all examples with the first sense. It is significantly worse on the sentence task, although it is the second best performing algorithm and several percent above DL-CoTrain on that task.

Figure 3 shows (all) three examples from the named entity test set where Yarowsky-prop-cautious θ-only is correct but none of the other Yarowsky variants are. Note that it succeeds despite misleading features; "maker" and "company' might be taken to indicate a company and "president" a person, but all three examples are locations.

Yarowsky-prop-cautious ϕ-θ and π-θ also perform respectably, although not as well. Yarowsky-prop-cautious θ^{T}-only and the non-cautious versions are significantly worse. Although θ^{T}-only was intended to incorporate Subramanya et al. (2010)'s idea of type level distributions, it in fact performs worse than θ-only. We believe that Collins and Singer (1999)'s definition (1) of θ incorporates sufficient type level information that the creation of a separate distribution is unnecessary in this case.

Figure 4 shows the test set non-seeded accuracies as a function of the iteration for many of the algo-

[7]The software is included with the paper submission and will be maintained at *https://github.com/sfu-natlang/yarowsky*.

Algorithm	Task							
	named entity		drug		land		sentence	
EM	81.05 $\pm$0.31	78.64 $\pm$0.34	55.96 $\pm$0.41	54.85 $\pm$0.43	32.86 $\pm$0.00	31.07 $\pm$0.00	67.88 $\pm$3.35	65.42 $\pm$3.57
Seed DL	*11.29*	*0.00*	*5.18*	*0.00*	*2.89*	*0.00*	*7.18*	*0.00*
DL-CoTrain (cautious)	91.56	90.49	**59.59**	**58.17**	78.36	77.72	68.16	65.69
Yarowsky	*81.19*	*78.79*	55.70	54.02	*79.03*	*78.41*	*62.91*	*60.04*
Yarowsky-cautious	91.11	89.97	54.40	52.63	***79.10***	***78.48***	***78.64***	***76.99***
Yarowsky-cautious sum	91.56	90.49	54.40	52.63	78.36	77.72	***78.64***	***76.99***
HS-bipartite avg-avg	*45.84*	*45.89*	*52.33*	*50.42*	78.36	77.72	*54.56*	*51.05*
HS-bipartite avg-maj	*81.98*	*79.69*	*52.07*	*50.14*	78.36	77.72	*55.15*	*51.67*
HS-bipartite maj-avg	*73.55*	*70.18*	*52.07*	*50.14*	78.36	77.72	*55.15*	*51.67*
HS-bipartite maj-maj	*73.66*	*70.31*	*52.07*	*50.14*	78.36	77.72	*55.15*	*51.67*
Yarowsky-prop ϕ-θ	*80.39*	*77.89*	*53.63*	*51.80*	78.36	77.72	*55.34*	*51.88*
Yarowsky-prop π-θ	*78.34*	*75.58*	54.15	52.35	78.36	77.72	*54.56*	*51.05*
Yarowsky-prop θ-only	*78.56*	*75.84*	54.66	52.91	78.36	77.72	*54.56*	*51.05*
Yarowsky-prop θ^{T}-only	*77.88*	*75.06*	*52.07*	*50.14*	78.36	77.72	*54.56*	*51.05*
Yarowsky-prop-cautious ϕ-θ	90.19	88.95	56.99	55.40	78.36	77.72	*74.17*	*72.18*
Yarowsky-prop-cautious π-θ	*89.40*	*88.05*	58.55	57.06	78.36	77.72	70.10	67.78
Yarowsky-prop-cautious θ-only	**92.47**	**91.52**	58.55	57.06	78.36	77.72	*75.15*	*73.22*
Yarowsky-prop-cautious θ^{T}-only	*78.45*	*75.71*	58.29	56.79	78.36	77.72	*54.56*	*51.05*
Num. train/test examples	89305 / 962		134 / 386		1604 / 1488		303 / 515	

Table 3: Test set percent accuracy and non-seeded test set percent accuracy (respectively) for the algorithms on all tasks. **Bold** items are a maximum in their column. *Italic* items have a statistically significant difference versus DL-CoTrain ($p < 0.05$ with a McNemar test). For EM, $\pm$ indicates one standard deviation but statistical significance was not measured.

rithms on the named entity task. The Yarowsky-prop non-cautious algorithms quickly converge to the final accuracy and are not shown. While the other algorithms (figure 4(a)) make a large accuracy improvement in the final retraining step, the Yarowsky-prop (figure 4(b)) algorithms reach comparable accuracies earlier and gain much less from retraining.

We did not implement Collins and Singer (1999)'s CoBoost; however, in their results it performs comparably to DL-CoTrain and Yarowsky-cautious. As with DL-CoTrain, CoBoost requires two views.

6.4 Cautiousness

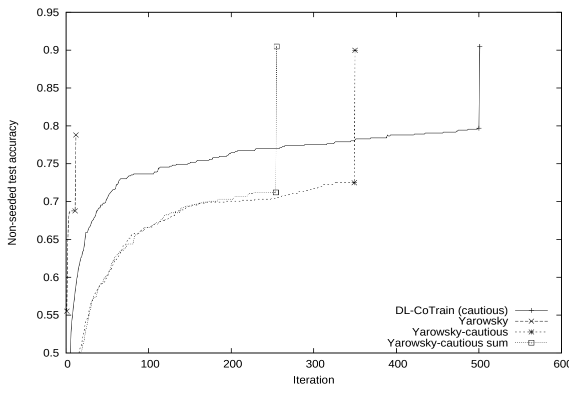
Cautiousness appears to be important in the performance of the algorithms we tested. In table 3, only the cautious algorithms are able to reach the 90% accuracy range.

To evaluate the effects of cautiousness we examine the Yarowsky-prop θ-only algorithm on the named entity task in more detail. This algorithm has two classifiers which are trained in conjunction: the DL and the propagated θ^{P}. Figure 5 shows the training set coverage (of the labelling on line 6 of algorithm 3) and the test set accuracy of both classifiers, for the cautious and non-cautious versions.

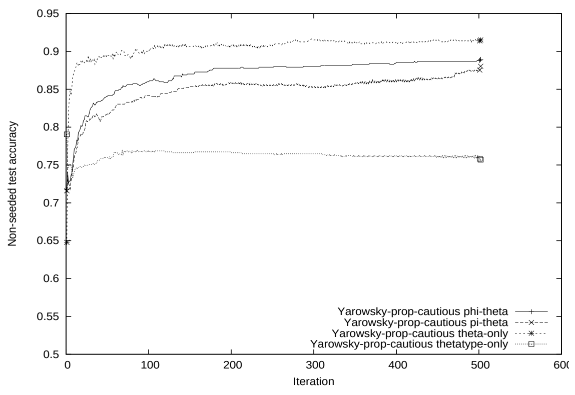
The non-cautious version immediately learns a DL over all feature-label pairs, and therefore has full coverage after the first iteration. The DL and θ^{P} converge to similar accuracies within a few more iterations, and the retraining step increases accuracy by less than one percent. On the other hand, the cautious version gradually increases the coverage over the iterations. The DL accuracy follows the coverage closely (similar to the behaviour of Yarowsky-cautious, not shown here), while the propagated classifier accuracy jumps quickly to near 90% and then increases only gradually.

Although the DL prior to retraining achieves a roughly similar accuracy in both versions, only the cautious version is able to reach the 90% accuracy range in the propagated classifier and retraining. Presumably the non-cautious version makes an early mistake, reaching a local minimum which it cannot escape. The cautious version avoids this by making only safe rule selection and labelling choices.

Figure 5(b) also helps to clarify the difference in retraining that we noted in section 6.3. Like the non-propagated DL algorithms, the DL component of Yarowsky-prop has much lower accuracy than the propagated classifier prior to the retraining step. But after retraining, the DL and θ^{P} reach very similar accuracies.



(a) Collins & Singer algorithms (plus sum form)



(b) Yarowsky propagation cautious

Figure 4: Non-seeded test accuracy versus iteration for various algorithms on named entity. The results for the Yarowsky-prop algorithms are for the propagated classifier θ^P, except for the final DL retraining iteration.

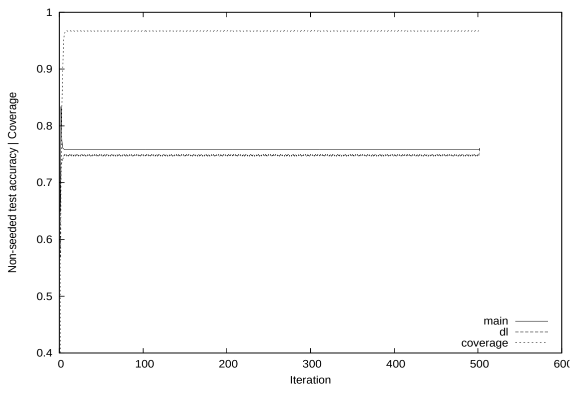
6.5 Objective function

The propagation method ϕ-θ was motivated by optimizing the equivalent objectives (2) and (3) at each iteration. Figure 6 shows the graph propagation objective (3) along with accuracy for Yarowsky-prop ϕ-θ without cautiousness. The objective value decreases as expected, and converges along with accuracy. Conversely, the cautious version (not shown here) does not clearly minimize the objective, since cautiousness limits the effect of the propagation.

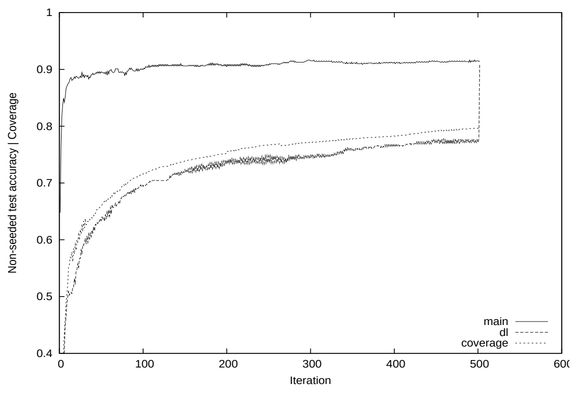
7 Conclusions

Our novel algorithm achieves accuracy comparable to Yarowsky-cautious, but is better theoretically motivated by combining ideas from Haffari and Sarkar (2007) and Subramanya et al. (2010). It also achieves accuracy comparable to DL-CoTrain, but does not require the features to be split into two independent views.

As future work, we would like to apply our algorithm to a structured task such as part of speech tagging. We also believe that our method for adapting Collins and Singer (1999)'s cautiousness to Yarowsky-prop can be applied to similar algorithms with other underlying classifiers, even to structured output models such as conditional random fields.



(a) Non-cautious



(b) Cautious

Figure 5: Internal train set coverage and non-seeded test accuracy (same scale) for Yarowsky-prop θ-only on named entity.

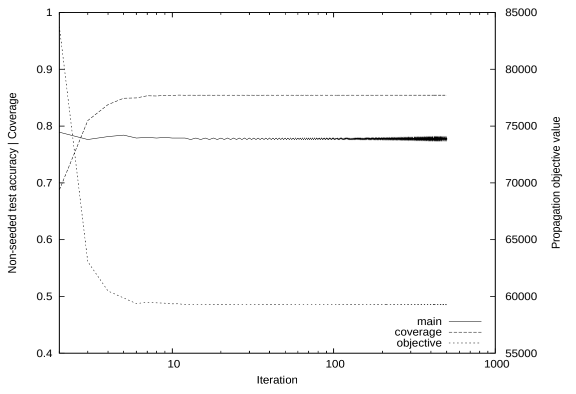


Figure 6: Non-seeded test accuracy (left axis), coverage (left axis, same scale), and objective value (right axis) for Yarowsky-prop ϕ-θ. Iterations are shown on a log scale. We omit the first iteration (where the DL contains only the seed rules) and start the plot at iteration 2 where there is a complete DL.

References

S. Abney. 2004. Understanding the Yarowsky algorithm. *Computational Linguistics*, 30(3).

A. Blum and S. Chawla. 2001. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 19th International Conference on Machine Learning (ICML-2001)*.

A. Blum and T. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of Computational Learning Theory*.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *In EMNLP 1999: Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain, July. Association for Computational Linguistics.

Hal Daume. 2011. Seeding, transduction, out-of-sample error and the Microsoft approach... Blog post at *http://nlpers.blogspot.com/2011/04/seeding-transduction-out-of-sample.html*, April 6.

Jason Eisner and Damianos Karakos. 2005. Bootstrapping without the boot. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 395–402, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Gholamreza Haffari and Anoop Sarkar. 2007. Analysis of semi-supervised learning with the Yarowsky algorithm. In *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada*, pages 159–166.

Aria Haghighi and Dan Klein. 2006a. Prototype-driven grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 881–888, Sydney, Australia, July. Association for Computational Linguistics.

Aria Haghighi and Dan Klein. 2006b. Prototype-driven learning for sequence models. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 320–327, New York City, USA, June. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 30(3).

H. J. Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11:363–371.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176, Cambridge, MA, October. Association for Computational Linguistics.

X. Zhu and Z. Ghahramani and J. Lafferty. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of International Conference on Machine Learning*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA, June. Association for Computational Linguistics.