BOOTSTRAPPING VIA GRAPH PROPAGATION

by

Max Whitney

B.Sc. (Computing Science), Simon Fraser University, 2009

A Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2012

APPROVAL

Name: Max Whitney

Degree: Master of Science

Title of Thesis: Bootstrapping via Graph Propagation

Examining Committee: Dr. Robert Hadley, Professor, School of Computing Science
Chair

Dr. Anoop Sarkar, Associate Professor, School of Computing Science
Senior Supervisor

Dr. Fred Popowich, Professor, School of Computing Science
Supervisor

Dr. Greg Mori, Associate Professor, School of Computing Science
Examiner

Date Approved: _____

Abstract

The Yarowsky algorithm is a simple self-training algorithm for bootstrapping learning from a small number of initial seed rules which has proven very effective in several natural language processing tasks. 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 thesis 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.

Keywords: natural language processing, computational linguistics, machine learning, unsupervised learning, semi-supervised learning, bootstrapping, graph propagation

Acknowledgments

Thanks are due first to my senior supervisor Dr. Anoop Sarkar, not only for being an excellent supervisor, but also for encouraging me to apply for graduate school after the undergraduate honours project which eventually led to this work.[1] Thanks to my supervisor Dr. Fred Popowich and my thesis examiner Dr. Greg Mori for reading and commenting on my thesis, and to Dr. Robert Hadley for chairing my defence. Thanks also to the rest of the natural language lab students for much help and motivation. Finally, thanks to my many other teachers (in computing science and elsewhere) as well as my family for getting me this far.

[1] Also for letting me write most of my research code in OCaml, and for providing me with a workstation named Darmok.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this thesis 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*. We focus on inductive rather than transductive methods.

The two dominant discriminative learning methods for bootstrapping are self-training (Scudder, 1965) and co-training (Blum and Mitchell, 1998). We focus on a self-training style bootstrapping algorithm, the Yarowsky algorithm of Yarowsky (1995), and particularly contrast it with the co-training algorithm DL-CoTrain of Collins and Singer (1999). Variants of the Yarowsky 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 ideas from the self-training (but not bootstrapping) algorithm of Subramanya et al. (2010), and introduce a novel algorithm called Yarowsky-prop which builds on the Yarowsky algorithm. 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. Along the way we implement and evaluate several other bootstrapping algorithms and examine the cautiousness introduced by Collins and Singer (1999). In the appendices we provide more details of experimental results and details of implementation and data processing which may be needed for replicating results.

Much of this material was previously published in greatly reduced form as Whitney and

Sarkar (2012). A few parts, particularly appendices B, C.1 and D, were previously written as an unpublished undergraduate honours project report. They are included here with updates for consistency with newer experiments.

1.1 Setting

1.1.1 Supervised, unsupervised, and semi-supervised learning

There are three general approaches to machine learning (Zhu and Goldberg, 2009). Many machine learning techniques involve *supervised learning*, where a large amount of labelled data is available and we wish to train a model to provide such labelling on previously unseen new data. At the other extreme is *unsupervised learning*, where a large amount of unlabelled data is available but no labels at all are provided. In the middle is *semi-supervised learning*, where a large amount of unlabelled data is available along with a smaller amount of labelled data or a simple model which can provide labels for a small number of examples. In the unsupervised and semi-supervised cases we again wish to train a model to provide labels on previously unseen new data.

In common natural language processing tasks labels are part of speech tags, word senses, phrase boundaries and types, representations of constituency trees, etc. In these tasks a large amount of unlabelled data is available from ordinary written documents, but this naturally produced data rarely includes labels. This kind of labelling requires experts, and even for experts is time consuming to produce. Moreover, the resources required to produce large labelled data sets of this kind are concentrated on particular domains in particular languages. Thus unsupervised and semi-supervised learning are important in natural language processing.

Although complete labelled data sets are difficult and time-consuming to produce, it is often possible for experts and in some cases even non-experts to produce a small number of labelled examples or a few reliable rules for labelling. For example, Eisner and Karakos (2005) find that in classifying two senses of the word "sentence", the words "reads" and "served" nearby in text are good indicators of the language sense and punishment sense respectively. This kind of limited labelling is much easier to produce, but provides labels only for a small part of the available data. Therefore semi-supervised learning is a practical approach to many natural language processing tasks.

Besides this immediate practical advantage, semi-supervised learning has at least two

more advantages. Firstly, it has cognitive appeal in that humans clearly can learn concepts without being given large numbers of classified examples, but at the same time do often learn from limited guidance (Zhu and Goldberg (2009) discusses semi-supervised learning in a cognitive context, and also touches on the usefulness for natural language data). Secondly, semi-supervised learning avoids the complications encountered with evaluating unsupervised learning, where it is necessary to decide which categories found in training correspond to which categories in the test data.

Semi-supervised learning is the approach that we focus on here. In our work the labelled data for semi-supervised learning is from *seed rules*, simple rules which are strongly indicative of a single label and which together constitute a simple classifier.[1]

1.1.2 Domain adaptation and bootstrapping

One common kind of semi-supervised learning is *domain adaptation* (Jiang, 2007). In this setting sufficient labelled data is available for supervised training in one domain (source of data with particular properties), and we wish to train a model which we can apply to another domain. Since the labelled source domain and unlabelled target domain have different properties, a model trained on the former will not in general be directly applicable to the later. For example, the Wall Street Journal is a common source of data since a large amount of text from it has been labelled with part of speech tags and constituency trees, but text from blog posts will contain many different words and usages, as will text from scientific papers.

In contrast, in *bootstrapping* we do not have a full labelled data set for any domain. Rather we have a large amount of unlabelled data, and a handful of useful facts which can be supplied by a human or by a reliable but inflexible source such a dictionary. From this starting point we wish to train a model which we can apply to the complete unlabelled data set, and to new data from the same domain. Bootstrapping is the setting that we consider here.

[1] A similar approach is *prototypes*, a small set of simple canonical examples for each label (Haghighi and Klein, 2006b,a). For example (Haghighi and Klein, 2006b), prototypes for part of speech tagging could be a list of a few words for each part of speech tag. Prototypes are not complete training examples in that they do not include context or features (for part of speech tagging, prototypes are single words rather than words with context). In general the idea of seed rules is to provide very high precision rules, while prototypes may be useful without high precision. Haghighi and Klein (2006b) extract their part of speech prototypes automatically from words that frequently occur with a particular label, but this requires some amount of labelled data.

1.1.3 Transductive and inductive learning

Transductive and inductive learning are two approaches to using data (Zhu and Goldberg, 2009). In *transductive learning* the training data is available when test data is to be labelled. For example, in graph based transductive approaches the (partially or fully labelled) training data and the unlabelled test data might be embedded in a single graph, and the learning process would consist of spreading labels from the training data to the test data over the connections in the graph. In this case it is not necessary to learn an explicit model separate from the training data.

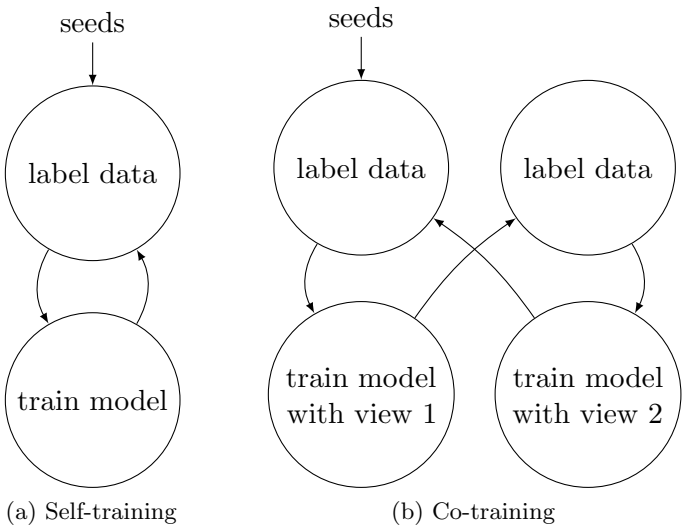
In *inductive learning* the training process creates a model which is based on the training data but separate from it. After this model is created there is no further need for the training data itself, and the model is applied independently to label previously unseen test data. In focusing on the Yarowsky algorithm and related algorithms which create an explicit model we limit ourselves to considering inductive learning here.

1.2 General approaches

One generative approach to semi-supervised learning is *clustering*. The examples are mapped to some feature space and a standard clustering algorithm is applied to group the examples in this space. The algorithm is given some set of expected labels for the examples, and after clustering these labels are assigned to the clusters found. However, it can be difficult to decide which labels belong to which clusters, especially if the number of clusters the algorithm finds is different from the expected number of labels (Castelli and Cover, 1995; Zhu, 2005).

Another generative approach is *maximum likelihood* estimation with the *EM* algorithm. EM is a powerful general approach, but requires a good generative model for a task and is sensitive to local optima (Merialdo, 1994; Johnson, 2007).

A simple discriminative approach is *self-training* (Scudder, 1965). In self-training a model is repeatedly trained on its own output, starting from some seed information in the form of a small model or a small amount of labelled data. The process is illustrated in fig. 1.1a. This approach often performs well in practice. However, it has little theoretical analysis or motivation. A popular self-training algorithm is the Yarowsky algorithm of Yarowsky (1995), which is the basis for our work.



(a) Self-training (b) Co-training

Figure 1.1: Outline of self-training and co-training.

Co-training (Blum and Mitchell, 1998) is a related discriminative approach. In co-training there are two classifiers which are alternately trained on each other's output, as in fig. 1.1b. Again the process starts from some seed information. The two classifiers may use the same model, but use different views (choices of features) for the data. This approach also performs well, and has good theoretical properties: PAC-learning analysis (Blum and Mitchell, 1998; Dasgupta et al., 2001) shows that in the case where the two views are conditionally independent of each other given a label, then when one trained classifier agrees with the other on unlabelled data it also has low generalization error. Co-training has been quite popular, with Blum and Mitchell (1998) receiving the 10-Year Best Paper Award at the 25th International Conference on Machine Learning (2008), and having a Google Scholar cited by count of 2471 at the time of this writing. However, the independence assumption on which the theoretical properties are based is unlikely to hold in real data (Abney, 2002).

1.3 The Yarowsky algorithm and its analysis

We focus on the *Yarowsky algorithm* and variants. The Yarowsky algorithm originates from Yarowsky (1995), and we particularly use the form from Collins and Singer (1999). It empirically performs well on various natural language processing tasks, but it lacks theoretical analysis. Collins and Singer (1999) introduce a feature called *cautiousness* which they apply

to the Yarowsky algorithm (and a related co-training algorithm), which they find to improve performance significantly.

Abney (2004) proposed several variants of the Yarowsky algorithm more susceptible to analysis and gave bounds for them. He does not provide empirical results and his algorithms are not cautious. Based on the importance of cautiousness in Collins and Singer (1999)'s results and our own, we do not believe that Abney (2004)'s algorithms would perform as well as Collins and Singer (1999)'s Yarowsky-cautious algorithm. However, his analysis is the basis for further work in formalizing the Yarowsky algorithm, including our own in this thesis.

Haffari and Sarkar (2007) expand on Abney (2004)'s analysis. They give further bounds, and introduce a view of bootstrapping as graph propagation on a bipartite feature-example graph. They also introduce a specific graph propagation algorithm based on the bipartite graph view which is a roughly a generalization of the Yarowsky algorithm. This algorithm is not cautious. They do not provide empirical results in the paper, but from personal communication and our own results this algorithm also does not perform as well as Collins and Singer (1999)'s Yarowsky-cautious. We also use this analysis in our own work.

1.4 Other work

There is a variety of work in other aspects of semi-supervised learning and bootstrapping. The graph-based approach of Haghighi and Klein (2006b) and Haghighi and Klein (2006a) also uses a small set of prototypes in a manner similar to 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. Zhu et al. (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). Large-scale information extraction, such as (Hearst, 1992), Snowball (Agichtein and Gravano, 2000), AutoSlog (Riloff and Shepherd, 1997), and Junto (Talukdar, 2010) among others, also have similarities to our approach. We focus on the formal analysis of the Yarowsky algorithm following Abney (2004).

1.5 Contributions

We make four particular contributions in this thesis:

1. We provide a cautious (in the sense of Collins and Singer (1999)), empirically well performing Yarowsky algorithm variant with a per-iteration objective function. This furthers the goals of Abney (2004) and Haffari and Sarkar (2007).

2. We integrate a variety of previous work on bootstrapping, especially Collins and Singer (1999), Abney (2004), Haffari and Sarkar (2007), and Subramanya et al. (2010).

3. We provide more evidence that cautiousness (Collins and Singer, 1999) is important, by showing results across several cautious and non-cautious algorithms.

4. We provide results from our implementations of several bootstrapping algorithms in addition to our own algorithm, allowing consistent comparison. We have made the software for these implementations available for others to replicate or expand on our results.[2]

1.6 Notation for bootstrapping

Abney (2004) defines useful notation for semi-supervised learning for labelling tasks, which we use here with minor extensions. This notation is shown in table 1.1. Here Y is a current labelling for the training data, as in self-training or co-training. Note that Λ, V, etc. are relative to the current Y.

Abney (2004) defines three probability distributions in his analysis of bootstrapping with the Yarowsky algorithm: θ_{fj} is the parameter for feature f with label j, taken to be normalized so that θ_f is a distribution over labels. This distribution represents the underlying model, especially the decision lists (section 3.1) used by the Yarowsky algorithm. ϕ_x is the labelling distribution over labels, and represents the current labelling 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, representing the classifier's current ideas about the probabilities of the possible labels for x. π_x does not have to be a point distribution. These distributions are summarized in table 1.2.

[2]Our source code is available at `https://github.com/sfu-natlang/yarowsky`.

x	denotes an example
f, g	denote features
j, 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
F	set of all features
J	set of all labels (not including $\perp$)
U	a uniform distribution

Table 1.1: Notation of Abney (2004) (above line) with our extensions (below line).

θ_f	parameter distribution for feature f
ϕ_x	labelling distribution for example x
π_x	prediction distribution for example x

Table 1.2: Distributions introduced by Abney (2004).

Chapter 2

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 seed rules we use for each task are also from these papers. They are decision lists as will be described in section 3.1, but they can also be thought of as simply strongly indicative features for each label. Appendix B discusses other ways to generate seed rules.

2.1 Named entity classification

The task of Collins and Singer (1999) is *named entity classification* (NE classification) on data from New York Times text. In this task each example is a noun phrase which is to be labelled as a person, organization, or location. The test set (but not the training set) additionally contains some noise examples which are not in any of the three named entity classes. Figure 2.1a shows the classes (labels) and the number of test set examples of each class.

Data set	Num. examples	
	Training	Testing
Named entity	89305	962
Drug	134	386
Land	1604	1488
Sentence	303	515

Table 2.1: Size of the data sets.

Label (named entity class)	Number of test examples
location	186
person	289
organization	402
noise	85

(a) named entity

Label (sense)	# test ex.
medical	189
illicit	197

(b) "drug"

Label (sense)	# test ex.
property	1157
country	331

(c) "land"

Label (sense)	# test ex.
judicial	258
grammatical	257

(d) "sentence"

Figure 2.1: Labels for each task and label counts in the test sets. The names for the senses in the word sense disambiguation tasks are from Gale et al. (1992).

They created their data as follows. The text was pre-processed with a statistical parser (Collins, 1997). Suitable noun phrases consisting of a sequence of proper nouns where the last word is the head of the phrase were extracted from the parse trees. In particular, either the phrase has an appositive modifier with a singular noun head word, or it is a complement to a preposition heading a prepositional phrase which modifiers another noun phrase with a singular noun head word. The actual words in the noun phrase being classified provide *spelling features*, while *context features* are extracted from the related appositive modifier or prepositional phrase. Figure 2.2 shows sample training data examples along with possible source sentences.[1]

The data as provided to us includes 89305 unlabelled training examples and 1000 labelled test examples. Following Collins and Singer (1999), we removed temporal items from the test set, leaving 962 testing examples (of which 85 are noise examples). We were unable to match the training set size described in the paper. See appendix D for more details on this issue and other aspects of using the data.

We use the seed rules the authors provide in the paper, which are shown in figure 2.3. Where there is ambiguity between the "Incorporated" and "Mr." rules we gave precedence

[1]These source sentences are from the Wall Street Journal corpus (WSJ88 as in Ratnaparkhi (1998)), where we found better matches than in the New York Times text which we have available (the processed named entity data as we received it does not include source sentences or any correspondences to source data). Perhaps there is text shared between the two data sets due to use of newswire text. In any case these sentences do illustrate how the features were created.

Source text sentence	Training example
" The practice of law in **Washington** is very different from what it is in Dubuque , " he said .	X0_Washington X11_law-in X3_RIGHT
MAI 's bid for Prime " makes no strategic sense , " said **Harvey L. Poppel** , a partner at **Broadview Associates** , an investment banker in Fort Lee , N.J . , that specializes in high technology mergers .	X0_Harvey-L.-Poppel X01_partner X2_Harvey X2_L. X2_Poppel X7_. X3_LEFT
	X0_Broadview-Associates X11_partner-at X2_Broadview X2_Associates X3_RIGHT

Figure 2.2: Three sample named entity training data examples and possible source sentences. The noun phrases to be classified are the X0_ features and the text marked in bold in the source. Features with prefixes X01_, X11_, and X3_ are context features, and all others are spelling features. See appendix D for the detailed feature types. Note that the second source sentence produces two data examples.

Feature	Label (named entity class)
X0_New-York	location
X0_California	location
X0_U.S.	location
X0_Microsoft	organization
X0_I.B.M.	organization
X2_Incorporated	organization
X2_Mr.	person

Figure 2.3: The seed rules for the named entity classification task. X0_ and X2_ features are both spelling features. X0_ features represent the whole phrase being classified, while X2_ features represent a single word the in phrase.

to the "Incorporated" rule; see appendix D.5. For co-training, we use their two views: one view is the spelling features, and the other is the context features.

2.2 Word sense disambiguation

The tasks of Eisner and Karakos (2005) are *word sense disambiguation* (WSD) on several English words which have two senses corresponding to two distinct words in French. We use their 'drug', 'land', and 'sentence' tasks (they discuss and provide data for other words, but do not provide seed rules). Figure 2.1 shows the senses (labels) and the number of test set examples of each sense.

They created the data as follows. Examples were extracted from the Canadian Hansards following Gale et al. (1992), using the English side to produce training and test data and the French side to produce the gold labelling for the test set. Features are the original and lemmatized words immediately adjacent to the word to be disambiguated, and original and lemmatized context words in the same sentence. The amount of data for each task is shown in table 2.1. Figure 2.4 shows sample training data examples with their apparent sources in the Hansards text (with data and sentences matched by hand since the data does not give correspondences to the source text; we use the ISI Hansards (ISI Hansards, 2001) for the source sentences).

We did no filtering to the data as we received it. The seed rules of Eisner and Karakos (2005) are pairs of adjacent word features, with one feature for each label (sense). They consider several different choices of seed rules. We use their reported best-performing seeds, which are shown in fig. 2.5. For co-training we use adjacent words for one view and context words for the other.

Note that the small training set sizes and relatively large test set sizes for these tasks may introduce sensitivities when learning on this data.[2] The "drug" task has especially little data, and has a test set much larger than its training set. The test set of the "sentence" task strongly favours the property sense; presumably the training set does as well, since we will see in results that most algorithms learn to label almost all examples with that sense and therefore usually end at the same accuracy.

2.3 Methodology and reporting results

Our goals are to replicate results for existing algorithms and to show equivalent performance for our novel algorithms. We use the same parameters reported in previous work when available. Since we use existing parameters and run only comparative experiments, we do not need a separate development set for independent tuning.

Collins and Singer (1999) define *clean accuracy* to be accuracy measured on only those test set examples which are not noise. Accuracy measured over all test set examples is then *noise accuracy*. Outside of the appendices we report only clean accuracy (see appendix A for the corresponding noise accuracies). There are no noise examples in the word sense

[2] In fact Eisner and Karakos (2005) appear to have chosen their data specifically to increase sensitivity to the choice of seed rules, which is their focus.

Source text sentence	Training example
He served part of his **sentence** and was released on parole .	PREVO_his PREVL_hi NEXTO_and NEXTL_and CONTO_served CONTL_serv CONTO_part CONTL_part CONTO_sentence CONTL_sentenc CONTO_released CONTL_releas CONTO_parole CONTL_parol
The next **sentence** reads :	PREVO_next PREVL_next NEXTO_reads NEXTL_read CONTO_sentence CONTL_sentenc CONTO_reads CONTL_read
The minimum penalty in either case , for either substance , is a suspended **sentence** .	PREVO_suspended PREVL_suspend NEXTO_. NEXTL_. CONTO_minimum CONTL_minimum CONTO_penalty CONTL_penalti CONTO_case CONTL_case CONTO_substance CONTL_substanc CONTO_suspended CONTL_suspend CONTO_sentence CONTL_sentenc
Then , a few **sentence**s later , he referred to the fact that in another few days there was a tremendous inflow of sums .	PREVO_the PREVL_the NEXTO_, NEXTL_, CONTO_publications CONTL_public CONTO_farmers CONTL_farmer CONTO_appears CONTL_appear CONTO_sentence CONTL_sentenc CONTO_truck CONTL_truck CONTO_hold CONTL_hold CONTO_tonnes CONTL_tonn

Figure 2.4: Four sample "sentence" word sense disambiguation training examples and possible source sentences. The NEXT[OL]_ features are the word immediately following the word to be disambiguated, PREV[OL]_ features are the word immediately preceding it, and CONT[OL]_ features are nearby context words. O features are the original (whole) words while L features are the lemmatized words.

Feature	Label (sense)
CONTO_medical	medical
CONTO_alcohol	illicit[a]

(a) "drug"

Feature	Label (sense)
CONTO_acres	property
CONTO_courts	country

(b) "land"

Feature	Label (sense)
CONTO_served	judicial
CONTO_reads	grammatical

(c) "sentence"

[a]In the data "alcohol" often occurs in the same sentence as a reference to illicit drugs even when the alcohol is not implied to be illicit itself.

Figure 2.5: The seed rules for the word sense disambiguation tasks.

disambiguation test sets, so clean and noise accuracy are equivalent on those tasks.

Daume (2011) suggests that when learning from seeds (or small amounts of initial information in other forms), accuracy should be measured on only those test set examples which cannot be labelled by the seeds. We call this kind of accuracy *non-seeded accuracy*. The intention is to measure the accuracy based on what is learned rather than what is given initially. We refer to standard (non-non-seeded) accuracy as *full accuracy*. We report both full and non-seeded accuracy, but usually focus on non-seeded accuracy in analysis and discussion.

Due to the small data set sizes for the word sense disambiguation tasks we focus on the named entity task when analyzing results.

When we report statistical significance we use a McNemar test and check that $p < 0.05$. When doing statistical significance testing on aggregate results (for EM; sections 4.2 and 4.7) we create each cell of the contingency table (counting where both of the algorithms being compared are correct, where one is correct and the other wrong, etc.) by taking the mean over repetitions and rounding to the nearest integer.

Chapter 3

The Yarowsky algorithm

Yarowsky (1995) introduces a simple self-training algorithm based on decision lists. In fact, he describes a general algorithm and a specific algorithm. The general algorithm is equivalent to what we call self-training here: any base supervised learner is repeatedly trained on its own output, starting from a small initial labelling from an external source. The specific algorithm uses this framework with a decision list base learner (Rivest, 1987; Yarowsky, 1994) for word sense disambiguation, with extra one-sense-per-discourse constraints optionally used to augment the labelling. Although he does not not use the term himself, subsequent work (Abney, 2004; Haffari and Sarkar, 2007) considers the *Yarowsky algorithm* to be self-training with decision lists, without necessarily including task-specific details such as the one-sense-per-discourse constraints.

Here we take Collins and Singer (1999)'s version of the Yarowsky algorithm as our reference form. The decision list-based algorithms start with a few initial seed rules rather than a small number of initial training set labels.

3.1 Decision lists

A *decision list* (DL) (Rivest, 1987) is an (ordered) list of *rules*, where each rule $f \to j$ consists of a feature f and a label j. The order is produced by assigning a score to each rule and sorting on this score.[1] To label an example, a decision list finds the highest ranked rule whose feature is a feature of the example, and uses that rule's label. If there is no such rule,

[1] Alternatively, the score can be considered part of the rule and the order is implicit.

it leaves the example unlabelled.

In Abney (2004)'s notation (see section 1.6), a decision list's labelling choices correspond to the prediction distribution

$$\pi_x(j) \propto \max_{f \in F_x} \theta_{fj}. \tag{3.1}$$

That this definition is given as a proportion because π_x is normalized to be a probability distribution over labels. Similarly, the parameters θ_{fj} are directly proportional to the scores assigned to decision list rules $f \to j$, but they are normalized so that θ_f is a distribution over labels whereas the scores themselves do not need to be. Representing the prediction and parameters as distributions is needed only for analysis and for algorithms which use them explicitly; the basic decision list model does not require such normalization. Nonetheless, in this work we often use θ to denote a decision list regardless of whether the normalized θ_f distributions are required.

For learning decision lists, Collins and Singer (1999) use the score

$$\frac{|\Lambda_{fj}| + \epsilon}{|\Lambda_f| + L\epsilon} \tag{3.2}$$

for rule $f \to j$, where ϵ is an additive smoothing constant. Therefore accounting for the possibility that some rules $f \to j$ may not be in the decision list we have

$$\theta_{fj} \propto \begin{cases} \frac{|\Lambda_{fj}|+\epsilon}{|\Lambda_f|+L\epsilon} & \text{if } f \to j \text{ is in the DL} \\ 0 & \text{otherwise.} \end{cases}$$

We use the scores (3.2) for the decision list algorithms in this thesis, and the normalized θ_{fj} values when needed (chapter 5). Figure 3.1 shows an example decision list. Here the top seven rules are initial seed rules with fixed scores, and the remaining rules have scores from (3.2). Figure 3.2 shows the result of applying this decision list to data.

The decision list scores sometimes produce ties when sorting the list. In our implementation, ties are broken by fixed orderings for the labels and features. Appendix C.1 discusses this issue in more detail.

3.2 Collins and Singer (1999)'s version of the Yarowsky algorithm

We use Collins and Singer (1999) for our exact specification of the Yarowsky algorithm. It is similar to the algorithm of Yarowsky (1995) but it omits the word sense disambiguation

Rank	Score	Feature	Label
1	0.999900	X0_New-York	location
2	0.999900	X0_California	location
3	0.999900	X0_U.S.	location
4	0.999900	X0_Microsoft	organization
5	0.999900	X0_I.B.M.	organization
6	0.999900	X2_Incorporated	organization
7	0.999900	X2_Mr.	person
8	0.999900	X0_California	location
9	0.999876	X01_analyst	person
10	0.999772	X2_Bank	organization
11	0.999765	X0_U.S.	location
12	0.999743	X2_National	organization
13	0.999733	X11_analyst-at	organization
14	0.999712	X2_International	organization
15	0.999612	X11_analyst-with	organization
		⋮	

Figure 3.1: A sample decision list, taken from iteration 5 of the Yarowsky algorithm on the named entity task. Scores are pre-normalized values from (3.2), not θ_{fj} values. Seed rules are indicated by **bold** ranks. See fig. 2.3 for more information on the seed rules, and appendix D for the other feature types.

Label	Example
⊥	X0_Steptoe X11_partner-at X3_RIGHT
1	**X0_California** X01_wholesaler X3_LEFT
3	X0_Republic-National-Bank X11_exchange-with X2_Republic X2_National **X2_Bank** X3_RIGHT
2	X0_Mr.-Suhud X01_minister **X2_Mr.** X2_Suhud X7_. X3_LEFT
⊥	X0_Japan-Indonesia X11_complexion-of X3_RIGHT
3	X0_P.T.-Astra-International-Incorporated X11_agreement-with X2_P.T. X2_Astra X2_International **X2_Incorporated** X7_.. X3_RIGHT
2	X0_Mr.-Buchanan X01_analyst **X2_Mr.** X2_Buchanan X7_. X3_LEFT
⊥	X0_Re X01_Failure X3_LEFT
⊥	X0_Kyocera X01_maker X3_LEFT
	⋮

Figure 3.2: Part of the named entity training data labelled by the decision list of fig. 3.1. **Bold** indicates the feature of the rule used to label each labelled example. ⊥ indicates examples which the decision list could not label and thus remain unlabelled.

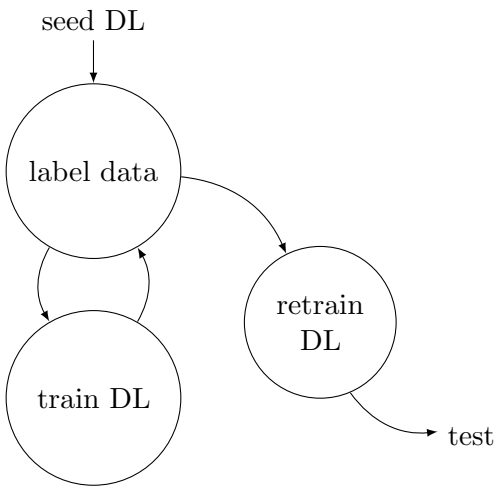


Figure 3.3: Outline of the Yarowsky algorithm as in Collins and Singer (1999).

optimization (one-sense-per-discourse constraints) and uses a different decision list scoring function. The algorithm is outlined in fig. 3.3 and shown in detail as algorithm 1. Here we use θ to denote the decision list, although the decision list scores do not need to be normalized for this algorithm.

Algorithm 1 The basic Yarowsky algorithm as in Collins and Singer (1999).

Require: training data X and a seed DL $\theta^{(0)}$
1: **for** iteration $t = 1, 2, \ldots$ to maximum or convergence **do**
2: apply $\theta^{(t-1)}$ to X to produce $Y^{(t)}$
3: train a new DL $\theta^{(t)}$ on $Y^{(t)}$ using (3.2), keeping only rules with score above ζ
4: **end for**
5: train a final DL θ on the last $Y^{(t)}$ using (3.2) *#retraining step*

Because of the threshold ζ on rule scores, the decision lists learned during training will not necessarily contain all possible rules and thus do not necessarily label the entire training set. Rather at each iteration the algorithm labels as much of the training data as is possible with the current decision list rules. However, after the training loop finishes the algorithm uses the last labelling to train a final decision list without using threshold, and this decision list is used for testing. We call this the *retraining step*.

In our implementation we add the seed rules to each subsequent decision list after applying the threshold, with fixed scores so that they are ranked near the top; this is not clearly specified in Collins and Singer (1999), but is used for the related DL-CoTrain algorithm (see

section 4.1) in the same paper.[2] We also add the seed rules to the final decision list after retraining, although it is not clear if Collins and Singer (1999) do this.

Note that a new labelling is created at each iteration. Therefore labels are not fixed between iterations. The label for an example may change, or a labelled example may become unlabelled. Convergence occurs if the labelling at one iteration is exactly the same as the labelling at the previous iteration (that is, $Y^{(t)} = Y^{(t-1)}$), since then the next learned decision list and all subsequent labellings and decision lists will be the same.

The Yarowsky algorithm has some similarity to EM and particularly hard EM, in that it alternately assigns labels to examples and then labels to features (in the decision list). However, the decision list model has advantages over simply counting and assigning probabilities. Especially, the decision list allows us to easily focus on high-quality rules, as we have already seen with threshold trimming. Since the decision list is not required to label all examples, we can remove parts of the model without concern. We return to this topic in section 6.5.

3.3 Collins and Singer (1999)'s Yarowsky-cautious algorithm

Collins and Singer (1999) also introduce a variant algorithm called Yarowsky-cautious. Cautiousness improves on the threshold for decision list trimming. The decision list learning step (line 3 of algorithm 1) keeps only the top n rules $f \to j$ over the threshold for each label j, ordered by $|\Lambda_f|$. Additionally the threshold ζ is checked against unsmoothed score $|\Lambda_{fj}|/|\Lambda_f|$ instead of the smoothed score (3.2) (also line 3). The decision list is still ordered by the smoothed scores after trimming. n begins at n_0 and is incremented by Δn at each iteration. We add the seed decision list to the new decision list after applying the threshold and cautious trimming. At the final iteration the Yarowsky-cautious algorithm does a retraining step without using either the threshold ζ or the cautiousness cutoff n, and again we add the seed rules.

Again some details in Collins and Singer (1999) are not clearly specified. In the specification above we made the following assumptions based on the description of decision list training for their DL-CoTrain algorithm:

[2] Yarowsky (1995) does not add the seed rules to subsequent decision lists. In his original algorithm seed rules may be overridden by rules learned during training, which he considers to be an advantage since it makes the algorithm less sensitive to bad seeds. In Collins and Singer (1999)'s Yarowsky algorithm as we understand it seed rules are almost certain to stay at the top of learned decision lists.

- the top n selection is per label rather in total
- the thresholding value is unsmoothed
- there is a retraining step
- the seed rules are added to each subsequent decision list
- seed rule adding is done after any threshold and cautiousness trimming is done.

We assume that in their description of DL-CoTrain the notation $Count'(x)$ is equivalent to their notation $Count(x)$ and therefore to Abney (2004)'s notation $|\Lambda_f|$. We also add the seed rules to the final decision list after retraining, although it is not clear if Collins and Singer (1999) do so. The orderings used when applying the thresholding and cautiousness cutoff also produces ties, which are discussed in appendix C.1.

3.4 Yarowsky-sum and Yarowsky-cautious-sum

Abney (2004) introduces several variants of the Yarowsky algorithm which are more suitable for formal analysis. We do not cover his variants in detail here since they are not cautious and we therefore believe that they would not perform as well as Yarowsky-cautious (he does not provide empirical results). However, we do consider his alternate prediction distribution in this section, and in section 4.4 we will consider another useful property of one variant.

For most of his Yarowsky algorithm variants, Abney (2004) uses the alternate prediction distribution

$$\pi_x(j) = \frac{1}{|F_x|} \sum_{f \in F_x} \theta_{fj} \tag{3.3}$$

which we refer to as the sum definition of π. This definition does not match a literal decision list but is easier to analyze.

As a baseline for the sum definition of π, we introduce the Yarowsky-sum algorithm. It is exactly 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 (Lafferty et al., 2001) for unstructured multiclass problems. Similarly Yarowsky-cautious-sum is exactly the same as Yarowsky-cautious except for using the sum definition of π in this way.

3.5 Results

Table 3.1 shows the test accuracy results from the algorithms discussed in this chapter.[3] We also include the seed rules as a decision list with no training and Collins and Singer (1999)'s single best label baseline, which labels all test examples with the most frequent label (which we select by looking at the test data). We take our parameters from Collins and Singer (1999) (see section 2.3 for methodology), using smoothing $\epsilon = 0.1$, a threshold $\zeta = 0.95$, and cautiousness parameters $n_0 = \Delta n = 5$ where applicable, and limiting the algorithms to a maximum of 500 iterations. When adding seed rules to subsequent decision lists we give them a weight of 0.9999 for the named entity task (as in Collins and Singer (1999)) and 1.0 for the word sense disambiguation tasks; in initial experiments the exact weight did not effect results.

With standard Yarowsky and Yarowsky-cautious, cautiousness improves the accuracy (statistically) significantly on the named entity classification task and the "sentence" word sense disambiguation task. On the "land" task the two are statistically equivalent, while on the "drug" task there is a significant decrease in accuracy (see section 2.2 for the sensitivity of two these tasks). The results for Yarowsky-sum versus Yarowsky-cautious-sum are similar. Yarowsky-sum significantly outperforms Yarowsky on the named entity task. On the "drug" and "sentence" tasks the two are statistically equivalent, while on the "land" task Yarowsky-sum is significantly worse. However, Yarowsky-cautious-sum differs significantly from Yarowsky-cautious only on the "land" task, where it is significantly worse (again, the "land" task is particularly sensitive). Overall we believe that cautiousness is an important improvement and that the sum definition algorithms represent a very viable alternative to the standard decision list definition.

[3] In Whitney and Sarkar (2012) we reported results from implementations where the unsmoothed score was used for thresholding in both Yarowsky and Yarowsky-cautious. In the results here we correctly use the smoothed score (3.2) for Yarowsky non-cautious and the unsmoothed score for Yarowsky-cautious, following Collins and Singer (1999). Similarly we maintain the difference for the non-cautious and cautious forms of DL-CoTrain (section 4.1). See appendix C.2 for an evaluation of the thresholding choice for these algorithms. Yarowsky-prop (chapter 5) uses unsmoothed thresholding for both non-cautious and cautious.

Algorithm	Task							
	named entity		drug		land		sentence	
Seed rules only	*11.29*	*0.00*	*5.18*	*0.00*	*2.89*	*0.00*	*7.18*	*0.00*
Single best label	*45.84*	*45.89*	*51.04*	50.14	*77.76*	*77.72*	*50.10*	*51.05*
Yarowsky	83.58	81.49	59.07	57.62	79.03	78.41	58.06	54.81
Yarowsky-cautious	*91.11*	*89.97*	*54.40*	*52.63*	79.10	78.48	*78.64*	*76.99*
Yarowsky-sum	*85.06*	*83.16*	60.36	59.00	*78.36*	*77.72*	58.06	54.81
Yarowsky-cautious-sum	*91.56*	*90.49*	*54.40*	*52.63*	*78.36*	*77.72*	*78.64*	*76.99*

Table 3.1: Percent clean accuracy (full and non-seeded respectively in each box) for the Yarowsky algorithm and variants, along with baselines. *Italics* indicate a statistically significant difference versus the basic Yarowsky algorithm. See appendix A for other accuracy measures and other statistical significances.

Chapter 4

Other bootstrapping algorithms

There are a variety of other bootstrapping algorithms. In this chapter we consider several. Some are useful as comparison for the Yarowsky variant algorithms that we focus on, while others provide useful ideas toward a novel algorithm.

4.1 DL-CoTrain

Collins and Singer (1999) introduce the co-training algorithm DL-CoTrain, which is shown as algorithm 2. This algorithm alternates between two decision lists using disjoint views of the features in the data. We represent the views simply as sets of features. At each step the algorithm trains a decision list and then produces a new labelling for the other decision list. At the end the decision lists are combined (and ordered together given their rule scores), the result is used to label the data, and a retraining step to learn a single decision list over all features is done from this single labelling.

Collins and Singer (1999) present this algorithm only with cautiousness. Here we use it both with and without for comparison. Without cautiousness, the learning for each decision list follows the same procedure described for Yarowsky in section 3.2. With cautiousness it follows the same procedure described for Yarowsky-cautious in section 3.3. Again we add the seed rules to the final decision list after retraining. In Collins and Singer (1999)'s description and our main experiments, all seed rules must use features from the first view. However, algorithm 2 shows a slight generalization which handles seed rules from both views, which we make use of in our experiments in appendix B.

Algorithm 2 Collins and Singer (1999)'s DL-CoTrain algorithm. Here the notation $\theta : V$ refers to the decision list θ filtered to contain only rules with features from view V. Note that the superscript expression $t - v'$ on line 4 is used because when labelling data for view 0 we use a decision list from the previous iteration, but when labelling data for view 1 we use a decision list from the current iteration. See fig. 1.1b for a flowchart of general co-training.

Require: training data X, two sets of features (views) V_0 and V_1, and a seed DL $\theta^{(0)}$
1: let $\theta^{(0,0)} = \theta^{(0)} : V_0$
2: **for** iteration $t = 1, 2, \ldots$ to maximum or convergence **do**
3: **for** view (v, v') in (0, 1), (1, 0) **do**
4: apply $\theta^{(t-v',v)}$ to X to produce $Y^{(t,v)}$
5: train a new DL $\theta^{(t,v')}$ on $Y^{(t,v)}$ using (3.2), using only features from $V_{v'}$ and keeping only rules with (smoothed or unsmoothed) score above ζ
6: add the rules from $\theta^{(0)} : V_{v'}$ to $\theta^{(t,v')}$
7: **end for**
8: **end for**
9: combine the last DLs $\theta^{(t,0)}$ and $\theta^{(t,1)}$ and apply the result to X to produce $Y^{(t)}$
10: train a final DL θ on $Y^{(t)}$ using (3.2), and add all rules from $\theta^{(0)}$ *#retraining step*

Collins and Singer (1999) also introduce an algorithm called CoBoost, which is a variant of co-training that they show to minimize an objective function at each iteration. We did not implement CoBoost, but in their results it performs comparably to DL-CoTrain and Yarowsky-cautious. As with DL-CoTrain and other co-training algorithms, CoBoost requires two views.

4.2 Collins and Singer (1999)'s EM algorithm

EM (Dempster et al., 1977) is a common approach to semi-supervised learning. Collins and Singer (1999) provide an EM algorithm which is a useful baseline for the other bootstrapping algorithms. Their generative model for EM uses probabilities $P(j)$, $P(|F_x|)$, and $P(f|j)$ so that the probability of a label j occurring with an example x is[1]

$$P(j, x) = P(j)P(|F_x|) \prod_{f \in F_x} P(f|j).$$

[1]They note that this model is deficient in that assigns non-zero probability to some impossible feature combinations.

The observed data for EM is the training data with a partial labelling Y given by the seed rules. The likelihood of the observed data is then

$$\prod_{x \in \Lambda} P(Y_x, x) \times \prod_{x \in V} \sum_{j \in J} P(j, x).$$

When labelling an example x they use $\arg\max_{j \in J} p(j, x)$.

Collins and Singer (1999) do not specify details of the algorithm besides doing random initialization, using EM with the model, and running to convergence. We implemented it as shown in algorithm 3. Note that here the labelling Y is from the seed rules only and does not change. To get accuracy comparable to Collins and Singer (1999)'s reported results we found it was necessary to include weights λ_1 and λ_2 on the expected counts of seed-labelled and unlabelled examples respectively (Nigam et al., 2000), and to use smoothing when calculating new probabilities. We randomly initialize all probabilities by choosing values from the interval $[1, 2]$ uniformly at random and then normalizing each distribution.

We extended the algorithm to optionally do hard EM and online EM (not shown in algorithm 3). For hard EM a fixed labelling (rather than a probability assignment) is produced at each E step and hard counts are taken on these labels (the sums in the E step of algorithm 3 become sums over only $x \in \Lambda$, with λ_2 being used as the weight for examples which are not labelled by the seeds). We added an optional threshold on $P(j, x)$ for labelling in hard EM, but did not find it to improve performance and do not include it in results here. For online EM the E and M steps are done separately for each training example in each iteration (adding an inner loop over examples in algorithm 3, with counts being updated incrementally).

4.3 Bregman divergences and ψ-entropy

Haffari and Sarkar (2007) use Bregman divergences and ψ-entropy notation (covered in more detail in Lafferty et al. (2001)) in their analysis of the Yarowsky algorithm. The Bregman divergence associated with ψ between two discrete probability distributions p and q is defined as

$$B_\psi(p, q) = \sum_i \left[\psi(p(i)) - \psi(q(i)) - \psi'(q(i))(p(i) - q(i))\right] \tag{4.1}$$

where the sum is over all i in the domain of p and q. Bregman divergence is a generalized divergence between distributions. The expression inside the sum in (4.1) represents the

Algorithm 3 Collins and Singer (1999)'s EM algorithm. $M = \{|F_x| \ : \ x \in X\}$ is the set of known example feature counts. Note that the labelling is fixed after line 1.

Require: training data X and a seed DL $\theta^{(0)}$

1: apply $\theta^{(0)}$ to X produce a labelling Y
2: randomize all $P(j)$, $P(m)$, and $P(f|j)$
3: **for** iteration t to maximum or convergence **do**
4: *#E step*
5: let $C(j) = \sum_{x \in \Lambda_j} \lambda_1 + \sum_{x \in V} \lambda_2 P(j, x)$
6: let $C(m) = \sum_{x \in \Lambda : |F_x| = m} \lambda_1 + \sum_{x \in V : |F_x| = m} \sum_{j \in J} \lambda_2 P(j, x)$
7: let $C(f, j) = \sum_{x \in \Lambda_{fj}} \lambda_1 + \sum_{x \in V_f} \lambda_2 P(j, x)$
8: *#M step*
9: let $P(j) = \frac{C(j)+\epsilon}{\sum_{k \in J}(C(k)+\epsilon)}$
10: let $P(m) = \frac{C(m)+\epsilon}{\sum_{n \in M}(C(n)+\epsilon)}$
11: let $P(f|j) = \frac{C(f,j)+\epsilon}{\sum_{g \in F}(C(g,j)+\epsilon)}$
12: **end for**

difference between the actual distance between $\psi(p(i))$ and $\psi(q(i))$, and the distance that would be expected based on linear growth after point $q(i)$. Figure 4.1 shows how a value i from the domain of the distributions contributes to the sum in (4.1).

ψ-entropy is defined as a generalization of Shannon entropy:

$$H_\psi(p) = -\sum_i \psi(p(i)).$$

Given this definition, Bregman divergence is analogous to KL-Divergence. Finally, ψ cross entropy can be defined in terms of entropy and divergence as in the standard definition:

$$H_\psi(p, q) = H_\psi(p) + B_\psi(p, q).$$

As a specific case we have $\psi = t^2$, which is the form that Haffari and Sarkar (2007) use and which we will use below. In this case we have

$$H_{t^2}(p) = -\sum_i p(i)^2$$

and Bregman divergence is just the mean-squared divergence (Euclidean norm squared):

$$B_{t^2}(p, q) = \sum_i (p(i) - q(i))^2 = ||p - q||^2.$$

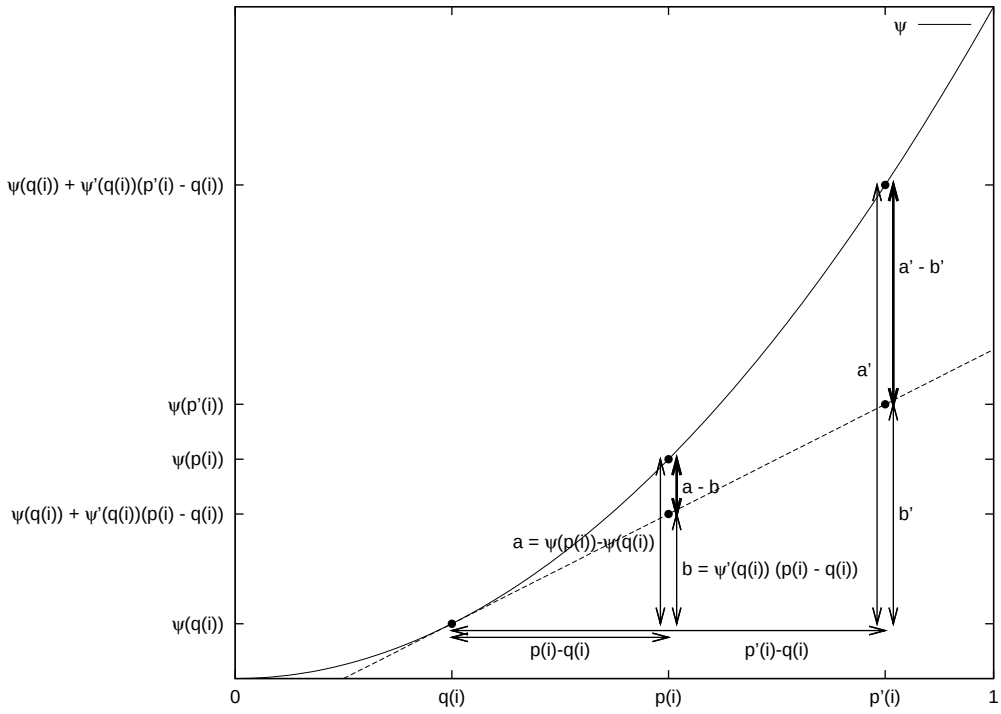


Figure 4.1: Illustration of the definition of Bregman divergence B_ψ. Here i is a fixed value from the domain of the distributions q, p, and p'. The horizontal axis is the probabilities under the distributions, and the vertical axis is these probabilities transformed by ψ. The length marked a is the actual difference between $\psi(p(i))$ and $\psi(q(i))$, while the length marked b is the expected difference based on a linear estimate of ψ from the tangent line at $(q(i), \psi(q(i)))$. The length marked $a - b$ is then the contribution from i to the Bregman divergence between p and q. The lengths marked a', b', and $a' - b'$ are the equivalent for the Bregman divergence between p' and q.

4.4 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 (3.2) is replaced by $1/|V_f|$. Secondly, π is redefined as the sum definition eq. (3.3).

We are not concerned here with the details of Y-1/DL-1-VS and we do not implement or test it here. However, we are interested in an objective function that Haffari and Sarkar (2007) provide for this algorithm, which will motivate our objective in section 5.1. They show that Y-1/DL-1-VS minimizes the following objective, which they write using t^2 cross entropy:

$$\begin{aligned}\sum_{\substack{x\in X\\ f\in F_x}} H_{t^2}(\phi_x,\theta_f) &= \sum_{\substack{x\in X\\ f\in F_x}} [H_{t^2}(\phi_x) + B_{t^2}(\phi_x,\theta_f)] \\ &= \sum_{\substack{x\in X\\ f\in F_x}} B_{t^2}(\phi_x,\theta_f) + |F_x| \sum_{x\in X} H_{t^2}(\phi_x). \end{aligned} \tag{4.2}$$

4.5 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 (4.2). The graph has vertices $F \cup X$ and edges $\{(f,x) \ : \ f \in F_x, x \in X\}$, as shown in figure 4.2. The feature vertices represent the parameter distributions θ while the example vertices represent the labelling distributions ϕ. In this view the Yarowsky algorithm can be seen as graph propagation which alternately updates the example distributions based on the feature distributions and visa versa.

Based on this they give algorithm 4, which we call HS-bipartite. It is parametrized by two functions which are called *examples-to-feature* and *features-to-example* here. Each function can be one of two choices: *average*(S) is the normalized average of the distributions in S, while *majority*(S) is a uniform distribution if all labels are supported by equal numbers of distributions in S, and otherwise a point distribution with mass on the label supported by the most distributions in S. The *average-majority* form (ie *examples-to-feature* = *average*, *features-to-example* = *majority*) is similar to Y-1/DL-1-VS,[2] and the *majority-majority* form

[2]It is not the same as Y-1/DL-1-VS in our implementation due to many details of the distribution updates

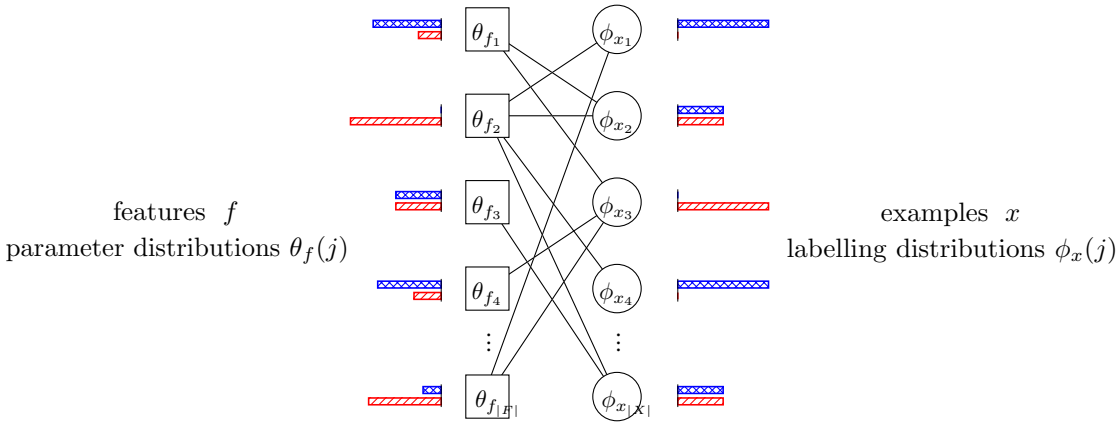


Figure 4.2: Haffari and Sarkar (2007)'s bipartite graph.

minimizes a different objective similar to (4.2).

Haffari and Sarkar (2007) do not specify details of applying the algorithm. 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 $\phi_x = \text{examples-to-feature}(F_x)$ for the unseen training examples x. Any ties in labelling or in *majority* are broken by our fixed order for labels (see section 3.1). We use the fixed order for features as the order in which to visit feature vertices, and the data order as the order in which to visit example vertices.

Algorithm 4 HS-bipartite.

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

and labelling.

4.6 CRF self-training algorithm of Subramanya et al. (2010)

Subramanya et al. (2010) give a self-training algorithm for semi-supervised part of speech tagging. Unlike the other algorithms we discuss, it is for domain adaptation with large amounts of labelled data rather than bootstrapping with a small number of seeds. Thus we will not be concerned with the details of the algorithm or the task, but it motivates our work firstly in providing the form of graph propagation which we will describe in more detail in section 5.1 and use in sections 5.2 and 5.3, and secondly in providing motivation for the structure of the novel algorithm that we present in section 5.3.

This algorithm, shown as algorithm 5, is a self-training algorithm which begins from an initial partial labelling and repeatedly trains a classifier on the labelling and then relabels the data. It uses a conditional random field (CRF) (Lafferty et al., 2001) as the underlying supervised learner. It uses labelled data $\mathcal{D}_l$ and unlabelled data $\mathcal{D}_u$ which in their setting are from different domains.

Algorithm 5 Subramanya et al. (2010)'s self-training algorithm. Here we do not use Abney (2004)'s notation. The various Λ values are CRF parameters, with superscripts (s) and (t) referring to the source and target domains respectively. Iterations are indexed by n.

labelled data $\mathcal{D}_l$, unlabelled data $\mathcal{D}_u$, and initial CRF parameters $\Lambda^{(0)}$
let $\Lambda^{(\mathrm{s})} = \text{crf-train}(\mathcal{D}_l, \Lambda^{(0)})$
let $\Lambda_0^{(\mathrm{t})} = \Lambda^{(\mathrm{s})}$
for iterations $n = 0, 1, \ldots$ to convergence **do**
  let $\{p\} = \text{posterior\_decode}(\mathcal{D}_u, \Lambda_n^{(\mathrm{t})})$
  let $\{q\} = \text{token\_to\_type}(\{p\})$
  let $\{\hat{q}\} = \text{graph\_propagate}(\{q\})$
  let $\mathcal{D}_u^{(1)} = \text{viterbi\_decode}(\{\hat{q}\}, \Lambda_n^{(\mathrm{t})})$
  let $\Lambda_{n+1}^{(\mathrm{t})} = \text{crf-train}(\mathcal{D}_l \cup \mathcal{D}_u^{(1)}, \Lambda_n^{(\mathrm{t})})$
end for

This algorithm differs significantly from the Yarowsky algorithm in ways beyond the difference in setting: First, it explicitly creates a type-level model using n-gram types rather than using only the n-gram token information.[3] They argue that using propagation over types allows the algorithm to enforce constraints and find similarities that basic self-training cannot.

Second, instead of only training a CRF at each iteration this algorithm also does a step of

[3] In section 6.5 we will see that this is in fact not much of a difference versus the Yarowsky algorithm.

immediate graph propagation between distributions over the type level model. The Yarowsky algorithm can also be viewed as graph propagation globally over iterations, as we saw in section 4.5. However, the algorithm of Subramanya et al. (2010) uses graph propagation as a sub-step within each iteration. Moreover, the form of graph propagation they use directly optimizes a well-defined objective function which we will return to in section 5.1.

4.7 Results

Table 4.1 shows the results for the algorithms that we implemented in this chapter. We take our parameters from the previous publications wherever possible (see section 2.3 for methodology). Again we use (where applicable) smoothing $\epsilon = 0.1$, a threshold $\zeta = 0.95$, and cautiousness parameters $n_0 = \Delta n = 5$, following Collins and Singer (1999). For EM we use weights $\lambda_1 = 0.98$, and $\lambda_2 = 0.02$, which were found in initial experiments to be best values for basic EM. We did not re-tune the parameters for hard EM and online EM. For each EM experiment we use 10 different random initializations and 10 different orders of the training data, chosen uniformly at random. We try each initialization with each training data order for a total of 100 repetitions (but the training data order only effects results for the online forms of EM), and report mean and standard deviation. We limit the algorithms to a maximum of 500 iterations.

DL-CoTrain non-cautious and some forms of EM sometimes outperform Yarowsky non-cautious, but only perform statistically equivalently to Yarowsky-cautious on the "drug" task (see section 2.2 for the sensitivity of this task). The HS-bipartite algorithms (all non-cautious) perform at best similarly to Yarowsky non-cautious. DL-CoTrain cautious performs statistically equivalent to Yarowsky-cautious on the named entity and "drug" tasks, and is significantly worse on the "land" and "sentence" tasks.

Algorithm	Task							
	named entity		drug		land		sentence	
Seed rules only	*11.29*	*0.00*	*5.18*	*0.00*	*2.89*	*0.00*	*7.18*	*0.00*
Single best label	*45.84*	*45.89*	51.04	50.14	*77.76*	*77.72*	*50.10*	*51.05*
Yarowsky	*83.58*	*81.49*	*59.07*	*57.62*	79.03	78.41	*58.06*	*54.81*
Yarowsky-cautious	91.11	89.97	54.40	52.63	**79.10**	**78.48**	**78.64**	**76.99**
DL-CoTrain (non-cautious)	*87.34*	*85.73*	60.10	58.73	*78.36*	*77.72*	*54.56*	*51.05*
DL-CoTrain (cautious)	91.56	90.49	59.59	58.17	*78.36*	*77.72*	*68.16*	*65.69*
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*
EM	*82.53*	*80.31*	53.76	52.49	*32.91*	*31.12*	*67.65*	*65.23*
±	0.31	0.34	0.27	0.28	0.03	0.03	3.31	3.55
Hard EM	*83.10*	*80.95*	54.15	52.91	*41.76*	*40.12*	*66.02*	*63.47*
±	2.24	2.53	0.70	0.74	13.05	13.39	5.99	6.37
Online EM	*85.71*	*83.89*	55.44	54.29	*46.36*	*45.00*	*59.08*	*56.25*
±	0.40	0.45	0.88	0.94	20.80	21.29	3.14	3.28
Hard online EM	*82.62*	*80.41*	55.67	54.54	*51.73*	*50.51*	*59.04*	*56.28*
±	0.61	0.68	0.97	1.03	22.50	23.02	3.49	3.56

Table 4.1: Percent clean accuracy (full and non-seeded respectively in each box) for the algorithms that we implemented in this chapter, with baselines and other algorithms for comparison. The row marked ± for each EM algorithm is percent standard deviation. *Italics* indicate a statistically significance difference versus Yarowsky-cautious (see section 2.3 for how we do significance testing on the aggregated EM results). See appendix A for other accuracy measures and other statistical significances.

Chapter 5

The Yarowsky-prop algorithm

In this chapter we introduce a novel algorithm which is based primarily on the Yarowsky and Yarowsky-cautious algorithms that we discussed in chapter 3, but also integrates ideas on graph propagation from some of the algorithms that we examined in chapter 4. This algorithm is cautious, empirically performs comparably to Yarowsky-cautious, and minimizes a well-defined objective function at each iteration.

5.1 Subramanya et al. (2010)'s graph propagation

Section 4.6 described the self-training algorithm of Subramanya et al. (2010). The graph propagation component of this algorithm is a general method for smoothing distributions attached to vertices of a graph, based on Bengio et al. (2006). Note that this graph propagation is independent of their specific graph structure, distributions, and self-training algorithm.

Here we present this graph propagation using Bregman divergences as described in section 4.3, and we omit the option to hold some of the distributions at fixed values (which would add an extra term to the objective). The objective is then

$$\mu \sum_{\substack{u \in \mathcal{V} \\ v \in \mathcal{N}_u}} w_{uv} B_{t^2}(q_u, q_v) + \nu \sum_{u \in V} B_{t^2}(q_u, U) \tag{5.1}$$

where $\mathcal{V}$ is a set of vertices; $\mathcal{N}_u$ is the neighbourhood of vertex $u \in V$; q_u is an initial distribution for each vertex $u \in V$ to be smoothed; and w_{uv} is a weight on the edge between $u \in \mathcal{V}$ and $v \in \mathcal{V}$ such that $w_{uv} = w_{vu}$. The first term of this objective brings distributions

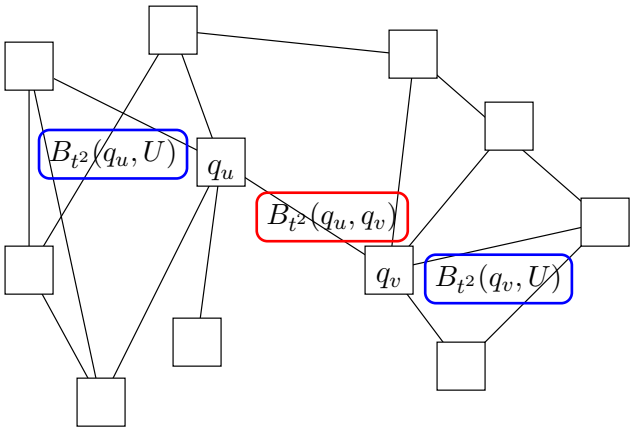


Figure 5.1: Visualization of Subramanya et al. (2010)'s graph propagation showing how the objective applies to two distributions q_u and q_v embedded in an abstract graph. The expression $B_{t^2}(q_u, q_v)$ is from the first term of (5.1) and brings q_u and q_v closer together. The expressions $B_{t^2}(q_u, U)$ and $B_{t^2}(q_v, U)$ are from the second term of (5.1) and are regularizers bringing each of q_u and q_v closer to uniform.

on neighbouring vertices closer together, while the second term is a normalizer bringing each distribution closer to uniform, as shown in fig. 5.1.

Subramanya et al. (2010) give an iterative update to minimize (5.1) under the constraint that all q_u must remain probability distributions: Let $q_u^{(0)} = q_u$ for all $u \in \mathcal{V}$. For $t > 0$, $q_u^{(t)}$ is given by

$$q_u^{(t)}(x) = \frac{\mu \sum_{v \in \mathcal{N}_u} w_{uv} q_v^{(t-1)}(x) + \nu U(x)}{\nu + \mu \sum_{v \in \mathcal{N}_u} w_{uv}}.$$

Subramanya et al. (2010) run to $t = 10$ iterations in their use of the updates.

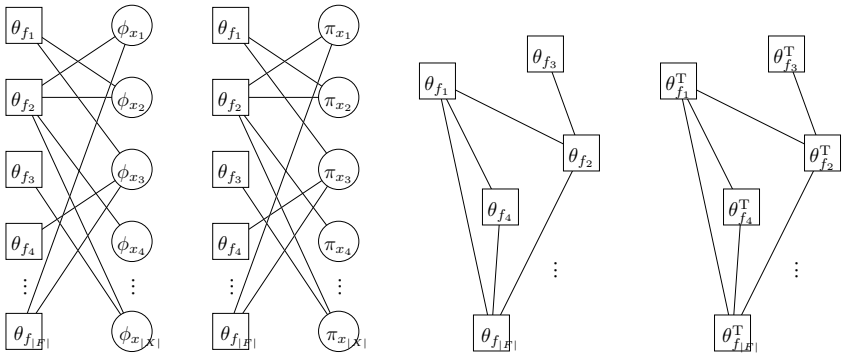
5.2 Applying graph propagation

Although the Yarowsky algorithm can be viewed as graph propagation, as we saw with HS-bipartite in section 4.5, the approach of Subramanya et al. (2010) differs in using graph propagation as a step at each iteration which adjusts a model produced by other means. Their approach also has the advantage of minimizing a well-defined objective function at each propagation step. We propose four methods for using this approach to graph propagation with the Yarowsky algorithm.

The distributions and graph structures for the four methods are shown in table 5.1, and

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 5.1: Graph structures for propagation.



(a) ϕ-θ method (b) π-θ method (c) θ-only method (d) θ^{T}-only method

Figure 5.2: Sample graphs for Yarowsky-prop propagation.

fig. 5.2 shows samples of the resulting graphs. In the θ^{T}-only method we define

$$\theta^{\mathrm{T}}_{fj} = \frac{1}{|X_f|} \sum_{x \in X_f} \pi_x(j)$$

using the sum definition (3.3) for π. θ^{T} is intended as a type-level version of θ. The graph structures of ϕ-θ and π-θ are the bipartite graph of Haffari and Sarkar (2007), with examples on one side, features on the other, and adjacency by features occurring in examples. The graph structures of θ-only and θ^{T}-only are unipartite with only the features as vertices, and adjacency by features co-occurring in the same example. Because adjacency in both the bipartite and unipartite graphs is defined by occurrences of features in examples, they are structurally similar; the bipartite graphs include example vertices connected between feature vertices, while in the unipartite graphs the feature vertices are connected to each other directly.

There are two sources of motivation for these methods. First, with the ϕ-θ method the objective (5.1) and Haffari and Sarkar (2007)'s Y-1/DL-1-VS objective (4.2) are very similar.

The regularization is different, but the first terms are identical up to constant coefficients.[1] Thus Subramanya et al. (2010)'s graph propagation with ϕ-θ gives us a direct way to optimize an objective similar to that for Y-1/DL-1-VS.

On the other hand, the unipartite graph methods θ-only and θ^{T}-only are more similar to the approach of the overall self-training algorithm of Subramanya et al. (2010), which does propagation on a unipartite graph of type-level features. The self-training algorithm of Subramanya et al. (2010) also extracts explicit type-level information from a separately learned token-level model. This is the reason to introduce θ^{T}-only. We will see in section 5.3 that with this method our algorithm closely resembles theirs.

5.3 Novel Yarowsky-prop algorithm

We call our novel algorithm Yarowsky-prop. It is shown with the different propagation methods as algorithms 6 to 9. Blank lines are added to the pseudocode in some cases to make shared parts appear on the same line numbers. The algorithms differ only at lines 3, 4 and 5. Yarowsky-prop is based on the Yarowsky algorithm, with the following changes: an added step of graph propagation to calculate an adjusted decision list θ^{P} (line 5), the use of θ^{P} rather than the decision list to update the labelling (line 6 and in testing), and the use of the sum definition of π (line 3 where used). Additionally with the ϕ-θ method the ϕ labelling distributions must be found explicitly, and with the θ^{T}-only method there is an additional step to calculate θ^{T} (both line 4). The decision list training step (line 7) is as we describe in sections 3.1 to 3.3.

Graph propagation (line 5) is done with the iterative updates of Subramanya et al. (2010). For the bipartite graph methods ϕ-θ and π-θ only the propagated θ values on the feature nodes are used for θ^{P}, and the distributions on the example nodes are ignored after the propagation step itself. Because we use Subramanya et al. (2010)'s graph propagation and iterative updates at each iteration, we know that Yarowsky-prop minimizes the objective function (5.1) at each iteration.

Although the algorithm constructs a decision list, when labelling data it uses θ^{P}_{fj} values

[1] The differences are specifically: First, (5.1) adds the constant coefficients μ and ν. Second, (5.1) sums over each edge twice (once in each direction), while (4.2) sums over each only once. Since $B_{t^2}(q_u, q_v) = B_{t^2}(q_v, q_u)$, as long as $w_{uv} = w_{vu}$ this can be folded into the constant μ. Finally, the regularization terms differ in sign and in constant factors.

Algorithm 6 Yarowsky-prop ϕ-θ.

Require: training data X and a seed DL $\theta^{(0)}$

1: let θ_{fj} be the scores of the seed rules *#crf_train*
2: **for** iteration t to maximum or convergence **do**
3:
4: label the data with θ to get labelling distribution ϕ
5: propagate on the ϕ-θ graph to get θ^{P} *#graph_propagate*
6: label the data with θ^{P} *#viterbi_decode*
7: learn a new DL θ_{fj} *#crf_train*
8: **end for**
9: train a final DL θ on the last labelling from θ^{P} *#retraining step*

Algorithm 7 Yarowsky-prop π-θ.

Require: training data X and a seed DL $\theta^{(0)}$

1: let θ_{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:
5: propagate on the π-θ graph to get θ^{P} *#graph_propagate*
6: label the data with θ^{P} *#viterbi_decode*
7: learn a new DL θ_{fj} *#crf_train*
8: **end for**
9: train a final DL θ on the last labelling from θ^{P} *#retraining step*

rather than the decision list itself.[2] The (pre-normalized) score for any rule $f \to j$ not in the decision list is taken to be zero. We use a sum of θ^{P}_{fj} values in labelling, as for calculating π: when labelling an example x 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 decision list behaviour of using evidence only from rules that are in the list. We use this method of labelling when applying θ^{P} to label both training and testing data, and similarly for the pre-propagated decision list θ when applying it to label training data (for cautiousness decisions and to calculate ϕ).

We include the thresholding of the Yarowsky algorithm (section 3.2) and optionally the cautiousness of Yarowsky-cautious (section 3.3) in learning a decision list θ (line 7). However, we check the threshold against unsmoothed rule scores (see sections 3.2 and 3.3)

[2] This is in contrast to the algorithms of Yarowsky (1995) and Collins and Singer (1999), which use the decision list directly for labelling.

Algorithm 8 Yarowsky-prop θ-only.

Require: training data X and a seed DL $\theta^{(0)}$

let θ_{fj} be the scores of the seed rules *#crf_train*
for iteration t to maximum or convergence **do**

  propagate on the θ-only graph to get θ^{P} *#graph_propagate*
  label the data with θ^{P} *#viterbi_decode*
  learn a new DL θ_{fj} *#crf_train*
end for
train a final DL θ on the last labelling from θ^{P} *#retraining step*

Algorithm 9 Yarowsky-prop θ^{T}-only.

Require: training data X and a seed DL $\theta^{(0)}$

let θ_{fj} be the scores of the seed rules *#crf_train*
for iteration t to maximum or convergence **do**
  let $\pi_x(j) = \frac{1}{|F_x|} \sum_{f \in F_x} \theta_{fj}$ *#post._decode*
  let $\theta^{\mathrm{T}}_{fj} = \frac{1}{|X_f|} \sum_{x \in X_f} \pi_x(j)$ *#token_to_type*
  propagate on the θ^{T}-only graph to get θ^{P} *#graph_propagate*
  label the data with θ^{P} *#viterbi_decode*
  learn a new DL θ_{fj} *#crf_train*
end for
train a final DL θ on the last labelling from θ^{P} *#retraining step*

both when using cautiousness and when not.[3] Additionally, when we label training data we label only examples which the pre-propagated decision list θ would also assign a label (using the θ_{fj} values for labelling, as with θ^{P}). That is, θ determines which examples to label and θ^{P} determines the labels themselves. We use this approach rather than only applying cautiousness to the decision list θ before propagation because the graph propagation tends to leave most θ^{P}_{f} non-uniform, so that cautiousness decisions in θ (rules missing in θ and thus many uniform θ_f values) would be disrupted in θ^{P}. Our approach ensures that the number of examples which are labelled at any iteration is the same as it would be without graph propagation. We preform a final retraining step where a conventional decision list is

[3] As opposed to Yarowsky and Yarowsky-cautious, where smoothed thresholding is used without cautiousness and unsmoothed thresholding is used with cautiousness. In initial experiments we observed unsmoothed thresholding to work better with Yarowsky-prop non-cautious in that it tends to produce higher accuracies and also at least with some graph structures seems to interfere less with the global behaviour of the propagation objective over iterations.

trained without thresholding or cautiousness on the training data labelling from θ^{P} at the last iteration.

There is similarity between this algorithm and the overall self-training algorithm of Subramanya et al. (2010), which can be seen by comparing algorithms 6 to 9 with algorithm 5. Comments in our algorithms give the corresponding parts of their algorithm. Note that each line except for the retraining step has a similar purpose. Algorithm 9 (θ^{P}-only) is most similar since it includes all the steps of their algorithm, including the token_to_type step.

5.4 Results

Table 5.2 shows the test accuracy results for the various forms of Yarowsky-prop. We take our parameters from previous publications wherever possible (see section 2.3 for methodology). Again we use (where applicable) smoothing $\epsilon = 0.1$, a threshold $\zeta = 0.95$, and cautiousness parameters $n_0 = \Delta n = 5$ as in Collins and Singer (1999), and we limit the algorithms to a maximum of 500 iterations. We use propagation parameters $\mu = 0.6, \nu = 0.01$ as in Subramanya et al. (2010) and uniform edge weights $w_{fx} = w_{fg} = 1$. Initial experiments with different μ and ν 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 settings for μ and ν that outperformed this choice. We experimented briefly with weighting edges by frequency of co-occurrences and did not observe any improvement in accuracy, but we have not yet properly evaluated this or other kinds of non-uniform edge weights and do not consider them in results here. We perform a single iteration of the graph propagation updates for each iteration of the Yarowsky-prop algorithm. Using more than one propagation update iteration per Yarowsky-prop iteration tends to decrease the final test accuracy; see tables A.8 and A.9 in appendix A.

The Yarowsky-prop algorithms without cautiousness perform at best similarly to Yarowsky non-cautious. With cautiousness the ϕ-θ, π-θ, and θ-only forms perform statistically equivalently to or significantly better than Yarowsky-cautious on the named entity and "drug" tasks. They are significantly below Yarowsky-cautious on the "land" task and "sentence" tasks (but see section 2.2 for why the "land" task is especially sensitive). The cautious θ^{T} form performs worse than the others on the named entity task, and is similar on the word sense tasks; see section 6.5 for discussion. Overall θ-only is the best performing form.

Algorithm	Task							
	named entity		drug		land		sentence	
Seed DL	*11.29*	*0.00*	*5.18*	*0.00*	*2.89*	*0.00*	*7.18*	*0.00*
Single best label	*45.84*	*45.89*	51.04	50.14	*77.76*	*77.72*	*50.10*	*51.05*
Yarowsky	*83.58*	*81.49*	*59.07*	*57.62*	79.03	78.41	*58.06*	*54.81*
Yarowsky-cautious	91.11	89.97	54.40	52.63	**79.10**	**78.48**	**78.64**	**76.99**
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*
Yar.-prop-cautious ϕ-θ	90.19	88.95	56.99	55.40	*78.36*	*77.72*	*74.17*	*72.18*
Yar.-prop-cautious π-θ	89.40	88.05	*58.55*	*57.06*	*78.36*	*77.72*	*70.10*	*67.78*
Yar.-prop-cautious θ-only	92.47	91.52	*58.55*	*57.06*	*78.36*	*77.72*	*75.15*	*73.22*
Yar.-prop-cautious θ^{T}-only	*78.45*	*75.71*	*58.29*	*56.79*	*78.36*	*77.72*	*54.56*	*51.05*

Table 5.2: Percent clean accuracy (full and non-seeded respectively in each box) for the different forms of the Yarowsky-prop algorithm, along with baselines and other algorithms included for comparison. *Italics* indicate a statistically significance difference versus Yarowsky-cautious. See appendix A for other accuracy measures and other statistical significances.

Chapter 6

Analysis

In this chapter we describe and analyze experimental results in more detail.

6.1 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). For EM we use weights $\lambda_1 = 0.98$, and $\lambda_2 = 0.02$. We let each algorithm run to a maximum of 500 iterations. See section 2.3 and the results sections in previous chapters for details on methodology and parameter choices. For EM results are averaged over 100 repetitions as described in section 4.7. See section 2.3 for information on clean versus noise accuracy, full versus non-seeded accuracy, statistical significance testing, and other details of how we report results.

6.2 Accuracy

The previous results sections (sections 3.5, 4.7 and 5.4) show the experimental clean test accuracies for each algorithm that we implemented. Appendix A shows combined test accuracy tables including both clean and noise accuracy, and also complete statistical significance information.

In the test accuracy results on the named entity task we see three general categories. We will refer to clean non-seeded accuracy here. HS-bipartite avg-avg performs worse than the other algorithms, at the same 45.89% accuracy of the single best label baseline. The

remaining non-cautious algorithms and the EM algorithms reach the 70% to mid 80% range, as does Yarowsky-prop-cautious θ^{T}-only. The remaining cautious algorithms reach the high 80% to low 90% range.

On the word sense disambiguation tasks (which we noted in section 2.2 to be sensitive due to small data sizes, especially for the "drug" and "land" tasks) the results are more mixed. On "drug" Yarowsky-sum non-cautious performs best with DL-CoTrain non-cautious and DL-CoTrain cautious next. Yarowsky non-cautious and Yarowsky-prop cautious with π-θ, θ-only, and θ^{T}-only are all close but are not (statistically) significantly better than the single best label baseline. The good performance of the Yarowsky-sum non-cautious and Yarowsky non-cautious algorithms on this task may be because using smoothed rule scores for thresholding (while the corresponding cautious algorithms use unsmoothed scores) is an advantage on the very small training set; see appendix C.2. On "land" most of the algorithms reach the same accuracy by giving almost all of the test examples a single label. Only Yarowsky and Yarowsky-cautious are significantly better than single best label, while the various forms of EM are significantly worse. On "sentence" Yarowsky-cautious and Yarowsky-cautious-sum perform best, with Yarowsky-prop-cautious θ-only and ϕ-θ next but significantly worse.

We focus more on the named entity task since the data sets are larger. We believe that overall these results support three conclusions concerning accuracy:

- Cautiousness is important.
- Yarowsky-cautious generally performs as well DL-CoTrain cautious and better than EM.
- Yarowsky-prop-cautious ϕ-θ and θ-only perform comparably to DL-CoTrain and Yarowsky-cautious.

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" word sense disambiguation task, is identical on the "land" task, 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 most examples with a single sense. It is significantly worse on the "sentence" task, although it is

Gold label	Spelling features	Context features
location	X0_Waukegan	X01_maker, X3_LEFT
location	X0_Mexico, X42_president, X42_of	X11_president-of, X3_RIGHT
location	X0_La-Jolla, X2_La, X2_Jolla	X01_company, X3_LEFT

Figure 6.1: Named entity test set examples where Yarowsky-prop θ-only is correct and no other tested algorithms are correct.

the next best performing algorithm after Yarowsky-cautious and Yarowsky-cautious-sum and several percent above DL-CoTrain on that task. 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 generally worse.

Figure 6.1 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-of" an organization, but all three examples are locations.

To the best of our knowledge Collins and Singer (1999) and Eisner and Karakos (2005) have the state of the art results on the named entity classification and word sense disambiguation tasks (with the specific data sets) respectively. Collins and Singer (1999) report a highest accuracy of 91.3% clean and 83.3% noise full accuracy on the named entity classification task (see table A.1 for their results with each algorithm). Our results are close to theirs with their own algorithms, and are numerically better with Yarowsky-prop-cautious θ-only. Eisner and Karakos (2005) report highest full accuracies of 76.1%, 81.3%, and 89.9% on the "drug", "land", and "sentence" word sense disambiguation tasks respectively (clean and noise accuracy are the same on these tasks), using a variant of the Yarowsky algorithm of Yarowsky (1995). Their results are intended to evaluate different methods of automatic seed selection, not different algorithms. Our best results are below theirs, especially on the "drug" and "sentence" tasks. This discrepancy may be due to the fact that we base our algorithms and parameters on Collins and Singer (1999), who developed them for the named entity task.

6.3 Decision list algorithm behaviour

Examination of how the decision list size, amount of labelled training data, and test accuracy interact shows why cautiousness is important and gives some sense of how Yarowsky-prop differs from Yarowsky-cautious in behaviour. Figure 6.2a shows the behaviour of Yarowsky

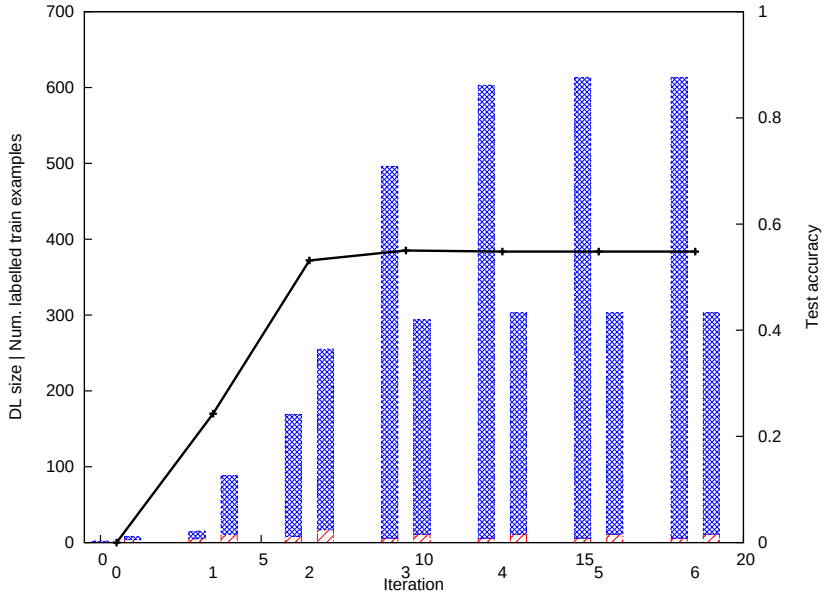
non-cautious on the "sentence" word sense disambiguation task. Recall that a decision list labels only those examples for which it has applicable rules, so that at any iteration of the algorithm it labels as many examples as possible but not necessarily all examples. In the figure the algorithm cannot label all of the 303 training set examples at iteration zero (with only the seed rules) or at iterations one (with the first learned decision list), two, or three. At iteration four and subsequent iterations it labels all training examples. It converges by iteration six. Note that labels are not fixed between iterations; although examples rarely become unlabelled after having been labelled, we can see in the figure that examples do change labels during the run. Also note that although the test accuracy increases rapidly in the first two iterations, it flattens out once most training examples have been labelled.

Figure 6.2b shows the behaviour of Yarowsky-cautious on the same task. In this case the decision list grows linearly and with roughly equal numbers of rules for each label (there are equal numbers of rules for each label among the learned rules, but not after adding the seeds). The number of labelled training examples and the test accuracy also grow gradually. Cautiousness limits not only the size of the decision list but also the number of labelled examples, prioritizing decisions which are believed to be of high accuracy. With cautiousness the algorithm avoids the case that trapped the Yarowsky non-cautious algorithm at low accuracy in fig. 6.2. The intuition here is that cautiousness, by making only high-confidence rule and labelling choices, avoids overeager labelling and early bad choices.

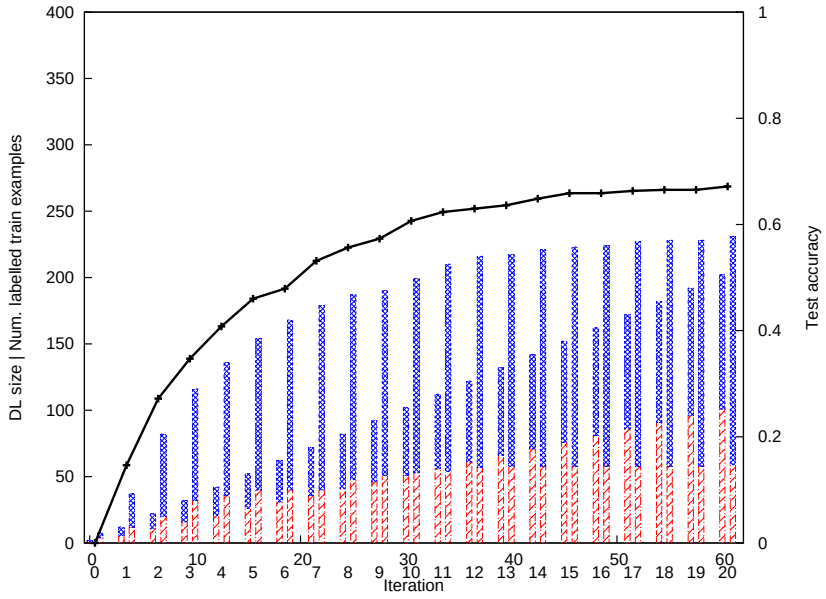
Figure 6.2c shows the behaviour of Yarowsky-prop-cautious in θ-only form on the same task. The training set labelling and test accuracies in this case are from the propagated classifier θ^{P}. Again the decision list grows linearly and with balanced labels due to cautiousness. In this case the amount of labelled training data and the test accuracy grow faster than with Yarowsky-cautious, but the algorithm still avoids the situation that trapped Yarowsky non-cautious. Although Yarowsky-cautious will pass Yarowsky-prop-cautious θ-only in accuracy at the retraining step (not shown in the figure) in this case, it is apparent that the graph propagation approach leads to more good early labellings than in Yarowsky-cautious.

6.4 Accuracy versus iteration

In section 6.3 we already saw test accuracy versus iteration for Yarowsky, Yarowsky-cautious, and Yarowsky-prop-cautious θ-only. Now we will look at more algorithms and examine the retraining step, this time on the named entity classification task. Figure 6.3 shows

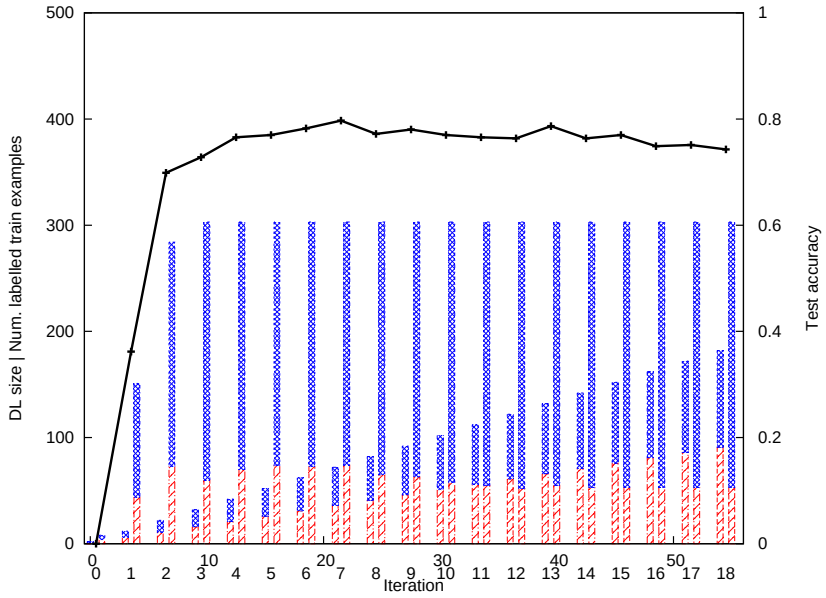


(a) Yarowsky



(b) Yarowsky-cautious

Figure 6.2: Behaviour over iterations of selected decision list-based algorithms on the "sentence" word sense disambiguation task. The final retraining step is not shown. The line shows test accuracy (right axis) of the current decision list. The left bar shows the current decision list size (left axis). The right bar shows the current number of labelled training set examples (left axis, same scale). Both bars are shaded by label. (Continued on page 46.)



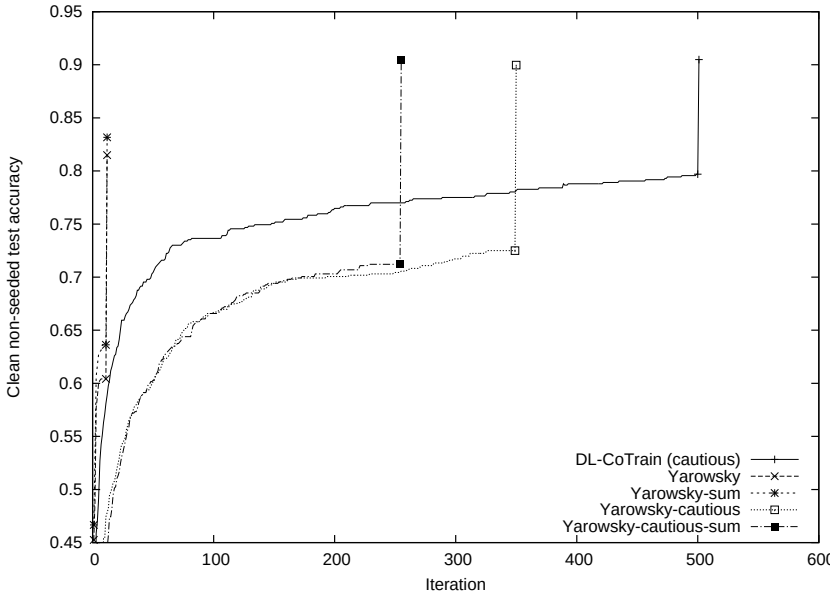
(c) Yarowsky-prop-cautious θ-only

Figure 6.2: Continued from page 45.

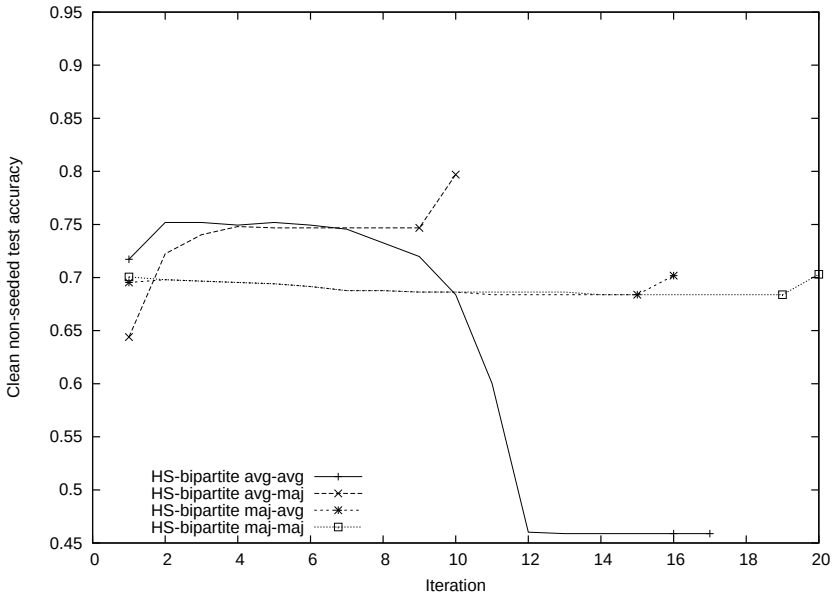
non-seeded test accuracy as a function of iteration for many of the algorithms.

We also look more closely at the accuracy over iterations for Yarowsky-prop. Yarowsky-prop has two classifiers which are trained in conjunction: the decision list and the propagated θ^{P}. Figure 6.4 shows the test accuracy versus iteration for both classifiers in the cautious and non-cautious versions of Yarowsky-prop θ-only, along with the training set coverage (of the labelling on line 6 of algorithms 6 to 9).

As we saw in section 6.3, in comparison to the other cautious decision list-based algorithms the Yarowsky-prop-cautious algorithms reach comparable accuracies earlier and gain much less from retraining. In figs. 6.3a and 6.3d we see that while the other algorithms make a large accuracy improvement in the retraining step, in Yarowsky-prop-cautious the propagated classifier has reached similar accuracies before retraining and gains only a little accuracy at retraining. Figure 6.4b clarifies this difference: like the non-propagated decision list algorithms, the decision list component of Yarowsky-prop-cautious has much lower accuracy than the propagated classifier prior to the retraining step. But after retraining (which is shown separately for each classifier in fig. 6.4), the decision list and θ^{P} reach very similar accuracies.

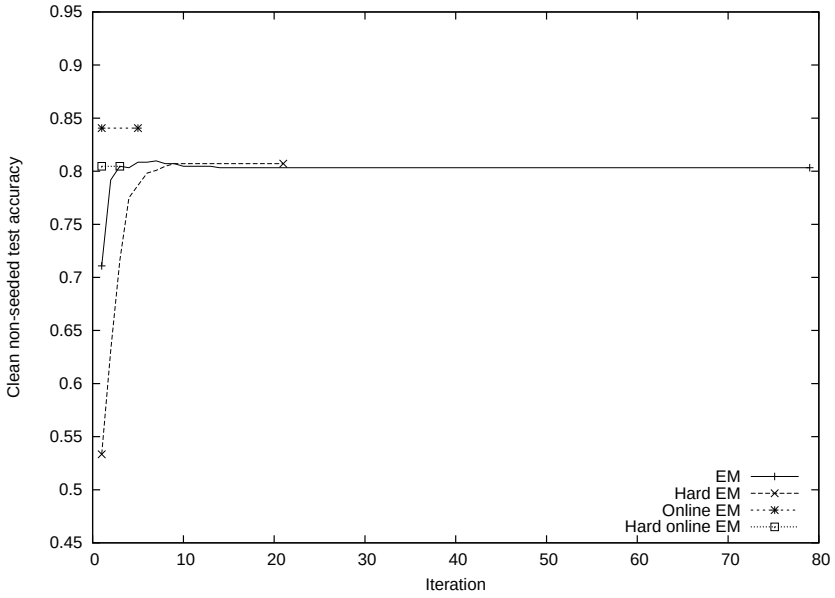


(a) Collins and Singer (1999) algorithms (plus sum forms)

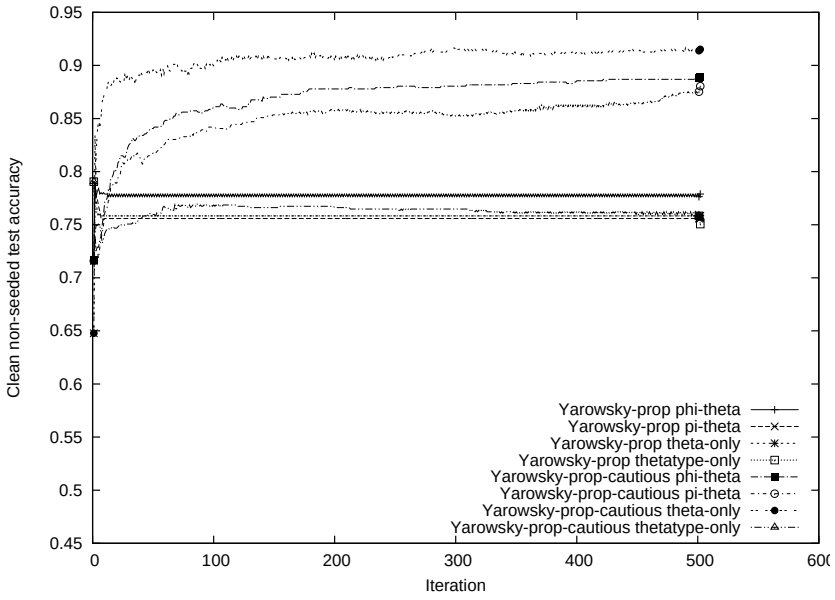


(b) HS-bipartite algorithms

Figure 6.3: Clean non-seeded test accuracy versus iteration for various algorithms on the named entity classification task. The results for the Yarowsky-prop algorithms are for the propagated classifier θ^P, except for the final retraining iteration where a normal decision list is used. For EM we show a single repetition (random initialization and training data order) for each variant, chosen to be roughly representative. (Continued on page 48.)

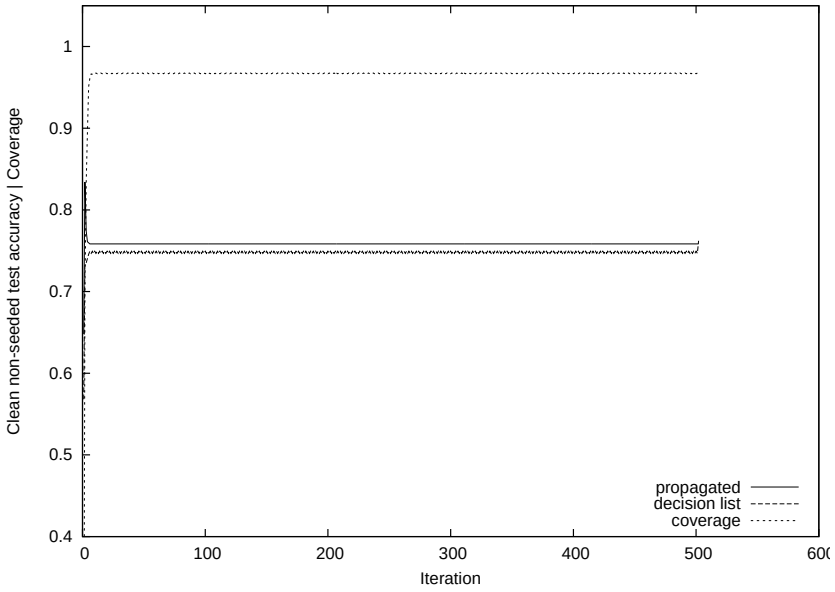


(c) EM algorithms (single repetitions)

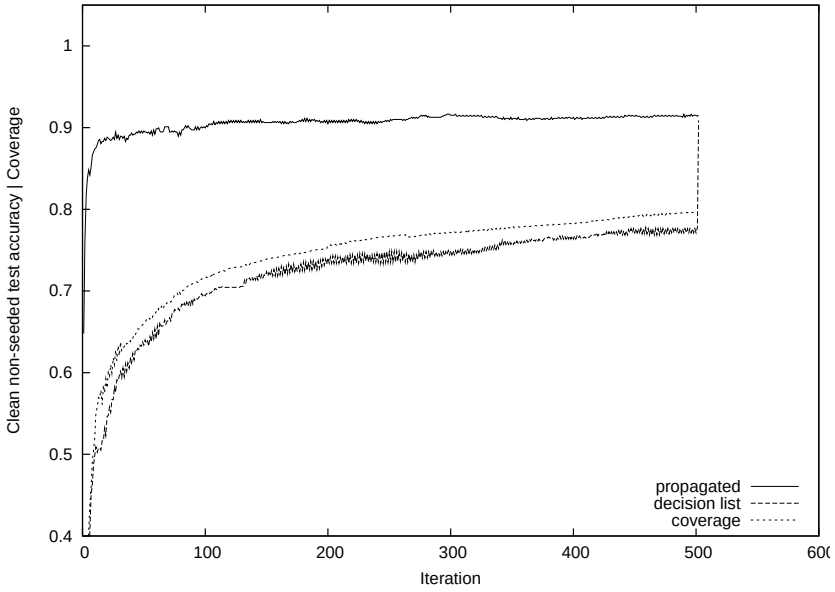


(d) Yarowsky-prop algorithms

Figure 6.3: Continued from page 47.



(a) Non-cautious



(b) Cautious

Figure 6.4: Clean non-seeded test accuracy for the base decision list θ and the propagated decision list θ^{P} for Yarowsky-prop θ-only on the named entity classification task, along with the internal training set coverage (same scale). Here at the retraining step we not only train a decision list on the last training set labelling from θ^{P}, but for comparison also label the data with the last base decision list θ and retrain a decision list for θ as well.

6.5 Type-level information and Yarowsky versus EM

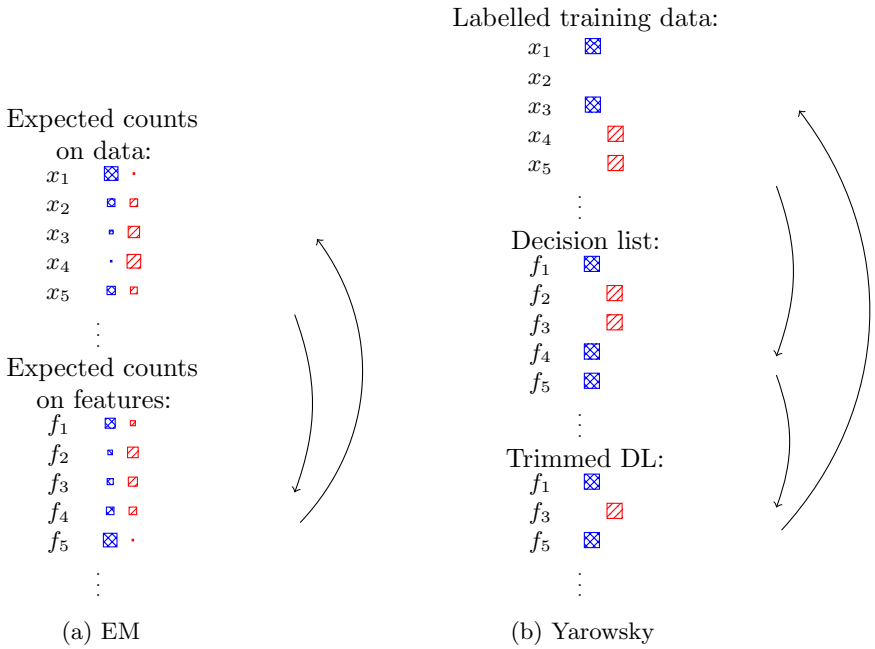
Although Yarowsky-prop θ^{T}-only was intended to incorporate Subramanya et al. (2010)'s idea of type level distributions, it in fact performs worse than θ-only and the bipartite graph forms of Yarowsky-prop. We believe that using Collins and Singer (1999)'s definition (3.2) for θ in fact incorporates sufficient type level information that the creation of a separate distribution is unnecessary in this case. Although the decision list does not explicitly represent different types of examples, the features are implicit categories. Moreover, the thresholding and cautiousness take advantage of this property by rejecting type level decisions (decision list rules) which are less certain.

This last property also bears on the difference between the Yarowsky algorithm and EM. The Yarowsky algorithm has some similarity to EM, which also alternately assigns (soft or hard) labels and then creates a model based on the labels. It is particularly like hard EM where concrete labels rather than probabilities are assigned. However, unlike EM the Yarowsky algorithm produces a model which represents not only expected labels but also types (rules). Again, this is particularly important since thresholding and cautiousness let us then improve the model. Figure 6.5 shows this difference.

6.6 Cautiousness

Cautiousness appears to be important in the performance of the algorithms we tested; in our results, only the cautious algorithms are able to reach the 90% test accuracy range on the named entity task. In section 6.3 we saw that in Yarowsky-cautious cautiousness allows the algorithm to escape cases that trap Yarowsky non-cautious at low accuracy. Comparing how Yarowsky-prop θ-only behaves without and with cautiousness in fig. 6.4a and fig. 6.4b respectively provides another look at the effects of cautiousness.

The non-cautious version immediately learns a decision list over all possible rules (feature-label pairs), and therefore has full coverage after the first iteration. The decision list and θ^{P} converge to similar accuracies within a few more iterations, and the retraining step increases test accuracy by less than one percent. On the other hand, the cautious version gradually increases the training set coverage over the iterations. The decision list test accuracy follows the training set coverage closely (similar to the behaviour of Yarowsky-cautious seen in fig. 6.2b), while the test accuracy of the propagated classifier quickly jumps to near 90% and



(a) EM (b) Yarowsky

Figure 6.5: Illustration of the differences between the Yarowsky algorithm and EM. Boxes represent assignments of labels (indicated by shade), which may be hard (a single choice as in the Yarowsky algorithm) or soft (probabilities as in non-hard EM).

then increases only gradually.

Although the decision list prior to retraining achieves a roughly similar test 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 situation where it cannot improve any further with immediately available decisions. The cautious version avoids this by making only safe rule selection and labelling choices.

6.7 Yarowsky-prop objective function

The propagation method ϕ-θ was motivated by optimizing the similar objectives (4.2) and (5.1) at each iteration. Figure 6.6 shows the graph propagation objective (5.1) along with accuracy for Yarowsky-prop ϕ-θ without cautiousness on the named entity classification task. 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.

6.8 Future work

There are a variety of ways to continue this research. Some of them are (ordered for convenience, not priority):

1. Apply the algorithms to more tasks and data sets (such as the SemEval tasks) for a more thorough comparison.

2. Apply the algorithms to structured tasks, especially part of speech tagging since it is a well-studied area.

3. Work on better integrating cautiousness into graph propagation approaches and their objective functions.

4. Work toward an algorithm which better integrates global graph propagation (as seen in Haffari and Sarkar (2007)'s analysis of the Yarowsky algorithm) with the per-iteration propagation used in Yarowsky-prop.

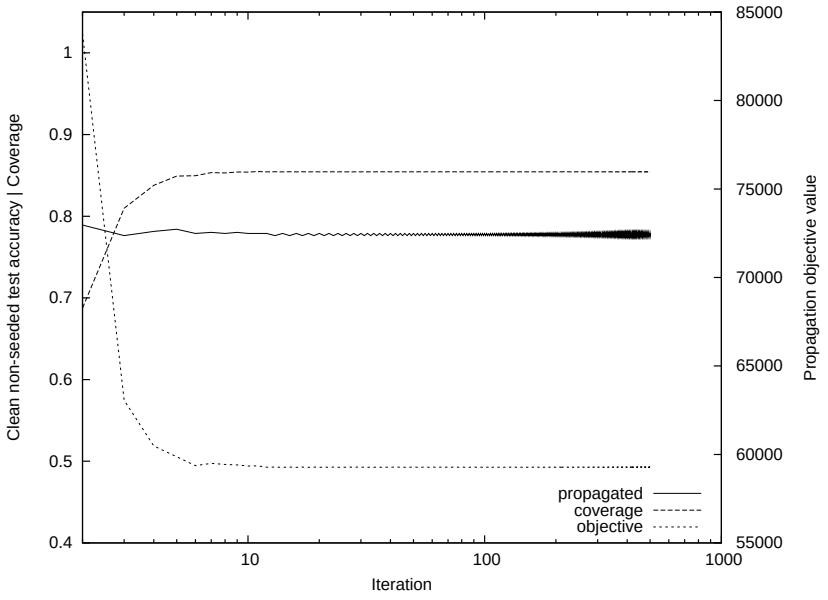


Figure 6.6: Clean non-seeded test accuracy (left axis), coverage (left axis, same scale), and graph propagation objective value (right axis) for Yarowsky-prop ϕ-θ non-cautious on the named entity classification task. Iterations are shown on a log scale. We omit the first iteration (where the decision list contains only the seed rules) and start the plot at iteration 2 where there is a complete decision list.

5. Collect more results on cautiousness, including alternate ways to choose rules, ways to integrate cautiousness into other algorithms, and ways in which cautiousness differs from other approaches such as thresholds on labelling certainty.

Of these, points 1 and 2 are natural extensions of the experiments in this work. Points 3 and 4 would refine the theoretical properties of our Yarowsky-prop algorithm, furthering the goal (from Abney (2004) and Haffari and Sarkar (2007)) of providing good theoretical analysis of self-training. Similarly point 5 would refine our understanding of cautiousness in bootstrapping.

Appendix A

Complete accuracy results

Here we show the complete accuracy results for our main experiments, and additional results for Yarowsky-prop. See section 2.3 for information on clean versus noise accuracy, full versus non-seeded accuracy, statistical significance testing, and other details of how we report results. Experimental settings are as described in section 6.1 and the results sections in previous chapters. In the results we include the seed rules as a decision list with no training and Collins and Singer (1999)'s single best label baseline, which labels all test examples with the most frequent label (which we select by looking at the test data).

Table A.1 shows the accuracy results reported by Collins and Singer (1999) for their algorithms. The remaining tables cover our own results. Tables A.2 and A.3 show the complete test accuracy results from our main experiments, along with statistical significance versus the Yarowsky-cautious algorithm. Tables A.4 to A.7 show pairwise statistical significance results on clean non-seeded accuracy for the same experiments. p values sometimes differ for other accuracy measurements (clean full, noise non-seeded, and noise full), while significance conclusions are usually the same but differ in a few indicated cases on the word sense tasks. We do not show significance on differences between pairs of EM results (due to the aggregation on them). Tables A.8 and A.9 show clean and noise accuracy results respectively for the various Yarowsky-prop algorithms as the number of iterations of graph propagation updates per Yarowsky-prop iteration is increased. In all other experiments (tables A.2 to A.7) we use one graph propagation update iteration per Yarowsky-prop iteration. Note that clean and noise accuracy differ only for the named entity task (where the test set includes noise examples).

Algorithm	Clean	Noise
Single best label	45.8	41.8
DL-CoTrain (cautious)	91.3	83.3
Yarowsky	81.3	74.1
Yarowsky-cautious	91.2	83.2

Table A.1: The full accuracies reported by Collins and Singer (1999) for those of their algorithms which we also evaluated, on the named entity classification task. They do not report non-seeded accuracy.

Algorithm	Task							
	named entity		drug		land		sentence	
Seed DL	*11.29*	*0.00*	*5.18*	*0.00*	*2.89*	*0.00*	*7.18*	*0.00*
Single best label	*45.84*	*45.89*	51.04	50.14	*77.76*	*77.72*	*50.10*	*51.05*
DL-CoTrain (non-cautious)	*87.34*	*85.73*	60.10	58.73	*78.36*	*77.72*	*54.56*	*51.05*
DL-CoTrain (cautious)	91.56	90.49	59.59	58.17	*78.36*	*77.72*	*68.16*	*65.69*
Yar.	*83.58*	*81.49*	*59.07*	*57.62*	79.03	78.41	*58.06*	*54.81*
Yar.-cautious	91.11	89.97	54.40	52.63	**79.10**	**78.48**	**78.64**	**76.99**
Yar.-sum	*85.06*	*83.16*	***60.36***	***59.00***	*78.36*	*77.72*	*58.06*	*54.81*
Yar.-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*
EM	*82.53*	*80.31*	53.76	52.49	*32.91*	*31.12*	*67.65*	*65.23*
±	0.31	0.34	0.27	0.28	0.03	0.03	3.31	3.55
Hard EM	*83.10*	*80.95*	54.15	52.91	*41.76*	*40.12*	*66.02*	*63.47*
±	2.24	2.53	0.70	0.74	13.05	13.39	5.99	6.37
Online EM	*85.71*	*83.89*	55.44	54.29	*46.36*	*45.00*	*59.08*	*56.25*
±	0.40	0.45	0.88	0.94	20.80	21.29	3.14	3.28
Hard online EM	*82.62*	*80.41*	55.67	54.54	*51.73*	*50.51*	*59.04*	*56.28*
±	0.61	0.68	0.97	1.03	22.50	23.02	3.49	3.56
Yar.-prop ϕ-θ	*80.39*	*77.89*	53.63	51.80	*78.36*	*77.72*	*55.34*	*51.88*
Yar.-prop π-θ	*78.34*	*75.58*	54.15	52.35	*78.36*	*77.72*	*54.56*	*51.05*
Yar.-prop θ-only	*78.56*	*75.84*	54.66	52.91	*78.36*	*77.72*	*54.56*	*51.05*
Yar.-prop θ^{T}-only	*77.88*	*75.06*	52.07	50.14	*78.36*	*77.72*	*54.56*	*51.05*
Yar.-prop-cautious ϕ-θ	90.19	88.95	56.99	55.40	*78.36*	*77.72*	*74.17*	*72.18*
Yar.-prop-cautious π-θ	89.40	88.05	*58.55*	*57.06*	*78.36*	*77.72*	*70.10*	*67.78*
Yar.-prop-cautious θ-only	**92.47**	**91.52**	*58.55*	*57.06*	*78.36*	*77.72*	*75.15*	*73.22*
Yar.-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 A.2: Percent clean accuracy (full and non-seeded respectively in each box). The rows marked ± for EM results are percent standard deviation. **Bold** indicates a maximum value in a column. *Italics* indicate a statistically significant difference versus Yarowsky-cautious.

Algorithm	Task							
	named entity		drug		land		sentence	
Seed DL	*10.29*	*0.00*	*5.18*	*0.00*	*2.89*	*0.00*	*7.18*	*0.00*
Single best label	*41.79*	*41.37*	51.04	50.14	*77.76*	*77.72*	*50.10*	*51.05*
DL-CoTrain (non-cautious)	*79.63*	*77.29*	60.10	58.73	*78.36*	*77.72*	*54.56*	*51.05*
DL-CoTrain (cautious)	83.47	81.58	59.59	58.17	*78.36*	*77.72*	*68.16*	*65.69*
Yar.	*76.20*	*73.46*	*59.07*	*57.62*	79.03	78.41	*58.06*	*54.81*
Yar.-cautious	83.06	81.11	54.40	52.63	**79.10**	**78.48**	**78.64**	**76.99**
Yar.-sum	*77.55*	*74.97*	***60.36***	***59.00***	*78.36*	*77.72*	*58.06*	*54.81*
Yar.-cautious-sum	83.47	81.58	54.40	52.63	*78.36*	*77.72*	**78.64**	**76.99**
HS-bipartite avg-avg	*41.79*	*41.37*	52.33	50.42	*78.36*	*77.72*	*54.56*	*51.05*
HS-bipartite avg-maj	*74.74*	*71.84*	52.07	50.14	*78.36*	*77.72*	*55.15*	*51.67*
HS-bipartite maj-avg	*67.05*	*63.27*	52.07	50.14	*78.36*	*77.72*	*55.15*	*51.67*
HS-bipartite maj-maj	*67.15*	*63.38*	52.07	50.14	*78.36*	*77.72*	*55.15*	*51.67*
EM	*75.24*	*72.40*	53.76	52.49	*32.91*	*31.12*	*67.65*	*65.23*
±	0.28	0.31	0.27	0.28	0.03	0.03	3.31	3.55
Hard EM	*75.76*	*72.98*	54.15	52.91	*41.76*	*40.12*	*66.02*	*63.47*
±	2.04	2.28	0.70	0.74	13.05	13.39	5.99	6.37
Online EM	*78.14*	*75.63*	55.44	54.29	*46.36*	*45.00*	*59.08*	*56.25*
±	0.37	0.41	0.88	0.94	20.80	21.29	3.14	3.28
Hard online EM	*75.32*	*72.49*	55.67	54.54	*51.73*	*50.51*	*59.04*	*56.28*
±	0.55	0.62	0.97	1.03	22.50	23.02	3.49	3.56
Yar.-prop ϕ-θ	*73.28*	*70.22*	53.63	51.80	*78.36*	*77.72*	*55.34*	*51.88*
Yar.-prop π-θ	*71.41*	*68.13*	54.15	52.35	*78.36*	*77.72*	*54.56*	*51.05*
Yar.-prop θ-only	*71.62*	*68.37*	54.66	52.91	*78.36*	*77.72*	*54.56*	*51.05*
Yar.-prop θ^{T}-only	*71.00*	*67.67*	52.07	50.14	*78.36*	*77.72*	*54.56*	*51.05*
Yar.-prop-cautious ϕ-θ	82.22	80.19	56.99	55.40	*78.36*	*77.72*	*74.17*	*72.18*
Yar.-prop-cautious π-θ	81.50	79.37	*58.55*	*57.06*	*78.36*	*77.72*	*70.10*	*67.78*
Yar.-prop-cautious θ-only	**84.30**	**82.50**	*58.55*	*57.06*	*78.36*	*77.72*	*75.15*	*73.22*
Yar.-prop-cautious θ^{T}-only	*71.52*	*68.25*	*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 A.3: Percent noise accuracy (full and non-seeded respectively in each box). The rows marked ± for EM results are percent standard deviation. **Bold** indicates a maximum value in a column. *Italics* indicate a statistically significant difference versus Yarowsky-cautious.

	Seed DL	Single best label	DL-CoTrain (non-cautious)	DL-CoTrain (cautious)	Yarowsky	Yarowsky-cautious	Yarowsky-sum	Yarowsky-cautious-sum	HS-bipartite avg-avg	HS-bipartite avg-maj	HS-bipartite maj-avg	HS-bipartite maj-maj	Yar.-prop ϕ-θ	Yar.-prop π-θ	Yar.-prop θ-only	Yar.-prop θ^{T}-only	Yar.-prop-cautious ϕ-θ	Yar.-prop-cautious π-θ	Yar.-prop-cautious θ-only	Yar.-prop-cautious θ^{T}-only
Seed DL	1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
Single best label	1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
DL-CoTrain (non-cautious)			1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.01**	.05	**.00**	**.00**
DL-CoTrain (cautious)				1.0	**.00**	.66	**.00**	1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	.11	**.01**	.24	**.00**
Yarowsky					1.0	**.00**	**.00**	**.00**	**.00**	.31	**.00**	**.00**	.57	.77	.10	.50	**.01**	**.00**	**.00**	**.00**
Yarowsky-cautious						1.0	**.00**	.39	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	.24	.05	.09
Yarowsky-sum							1.0	**.00**	**.00**	**.03**	**.00**	**.00**	.12	.20	.63	**.03**	**.00**	**.00**	**.00**	**.00**
Yarowsky-cautious-sum								1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	.06	**.01**	.23
HS-bipartite avg-avg									1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
HS-bipartite avg-maj										1.0	**.00**	**.00**	.74	.45	.00	.67	.31	**.03**	**.04**	**.01**
HS-bipartite maj-avg											1.0	1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
HS-bipartite maj-maj												1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
Yar.-prop ϕ-θ													1.0	**.03**	.07	**.02**	**.00**	**.00**	**.00**	.06
Yar.-prop π-θ														1.0	.69	.39	**.00**	**.00**	**.00**	1.0
Yar.-prop θ-only															1.0	**.03**	**.00**	**.00**	**.00**	1.0
Yar.-prop θ^{T}-only																1.0	**.00**	**.00**	**.00**	.27
Yar.-prop-cautious ϕ-θ																	1.0	.28	**.01**	**.00**
Yar.-prop-cautious π-θ																		1.0	**.00**	**.00**
Yar.-prop-cautious θ-only																			1.0	**.00**
Yar.-prop-cautious θ^{T}-only																				1.0

Table A.4: Statistical significance on the named entity classification task, showing p values for differences in clean non-seeded accuracy. **Bold** values indicate differences which are significant at $p < 0.05$. p values and significance conclusions are the same for other measures of accuracy (clean full, noise non-seeded, noise full) on this task.

	Seed DL	Single best label	DL-CoTrain (non-cautious)	DL-CoTrain (cautious)	Yarowsky	Yarowsky-cautious	Yarowsky-sum	Yarowsky-cautious-sum	HS-bipartite avg-avg	HS-bipartite avg-maj	HS-bipartite maj-avg	HS-bipartite maj-maj	EM	Hard EM	Online EM	Hard online EM	Yar.-prop ϕ-θ	Yar.-prop π-θ	Yar.-prop θ-only	Yar.-prop θ^{T}-only	Yar.-prop-cautious ϕ-θ	Yar.-prop-cautious π-θ	Yar.-prop-cautious θ-only	Yar.-prop-cautious θ^{T}-only
Seed DL	1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
Single best label	1.0	**.01**	**.02**	.62	.06†	.62	**.02**	.62	1.0	1.0	1.0	1.0	.66	.57	.33	.33	.62	.32	.16	1.0	.26	.12	.12	.15
DL-CoTrain (non-cautious)			1.0	.84	.73	.06	1.0	.06	**.01**	**.01**	**.01**	**.01**	.05†	.08	.18	.18	**.00**	**.02**	**.05**	**.01**	.33	.63	.62	.59
DL-CoTrain (cautious)				1.0	.91	.09	.82	.09	**.03**	**.02**	**.02**	**.02**	.07†	.10	.23	.27	**.00**	.05	.09	**.02**	.43	.78	.78	.72
Yarowsky					1.0	**.04**	.40	**.04**	.07	.06	.06	.06	.08	.11	.29	.33	**.05**	.14	.20	.06	.37	.88	.89	.80
Yarowsky-cautious						1.0	**.01**	1.0	.66	.62	.62	.62	1.0	1.0	.62	.55	.87	1.0	1.0	.62	.10	**.02**	**.02**	**.04**
Yarowsky-sum							1.0	**.01**	**.03**	**.02**	**.02**	**.02**	**.02**	**.04**	.10	.14	**.01**	.05	.08	**.02**	.17	.41	.40	.40
Yarowsky-cautious-sum								1.0	.66	.62	.62	.62	1.0	1.0	.62	.55	.87	1.0	1.0	.62	.10	**.02**	**.02**	**.04**
HS-bipartite avg-avg									1.0	1.0	1.0	1.0	.70	.61	.40	.36	.69	.39	.21	1.0	.29	.13	.13	.17
HS-bipartite avg-maj										1.0	1.0	1.0	.66	.57	.33	.33	.62	.32	.16	1.0	.26	.12	.12	.15
HS-bipartite maj-avg											1.0	1.0	.66	.57	.33	.33	.62	.32	.16	1.0	.26	.12	.12	.15
HS-bipartite maj-maj												1.0	.66	.57	.33	.33	.62	.32	.16	1.0	.26	.12	.12	.15
Yar.-prop ϕ-θ																	1.0	.89	.70	.62	.34	.12	.12	.17
Yar.-prop π-θ																		1.0	.73	.32	.49	.24	.24	.30
Yar.-prop θ-only																			1.0	.16	.59	.31	.31	.37
Yar.-prop θ^{T}-only																				1.0	.26	.12	.12	.15
Yar.-prop-cautious ϕ-θ																					1.0	.31	.33	.46
Yar.-prop-cautious π-θ																						1.0	1.0	1.0
Yar.-prop-cautious θ-only																							1.0	1.0
Yar.-prop-cautious θ^{T}-only																								1.0

Table A.5: Statistical significance on the "drug" WSD task, showing p values for differences in non-seeded accuracy (clean and noise accuracy are the same on this task). **Bold** values indicate differences which are significant at $p < 0.05$. † indicates differences which are not significant in non-seeded accuracy but are significant in full accuracy; all other signifances (but not necessarily p values) are the same for both accuracy measures.

	Seed DL	Single best label	DL-CoTrain (non-cautious)	DL-CoTrain (cautious)	Yarowsky	Yarowsky-cautious	Yarowsky-sum	Yarowsky-cautious-sum	HS-bipartite avg-avg	HS-bipartite avg-maj	HS-bipartite maj-avg	HS-bipartite maj-maj	EM	Hard EM	Online EM	Hard online EM	Yar.-prop ϕ-θ	Yar.-prop π-θ	Yar.-prop θ-only	Yar.-prop θ^{T}-only	Yar.-prop-cautious ϕ-θ	Yar.-prop-cautious π-θ	Yar.-prop-cautious θ-only	Yar.-prop-cautious θ^{T}-only
Seed DL		1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
Single best label	1.0		1.0†	1.0†	**.00**	**.04**	1.0†	1.0†	1.0†	1.0†	1.0†	1.0†	**.00**	**.00**	**.00**	**.00**	1.0†	1.0†	1.0†	1.0†	1.0†	1.0†	1.0†	1.0†
DL-CoTrain (non-cautious)	1.0	1.0			**.00**	**.04**	1.0	1.0	1.0	1.0	1.0	1.0	**.00**	**.00**	**.00**	**.00**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DL-CoTrain (cautious)		1.0			**.00**	**.04**	1.0	1.0	1.0	1.0	1.0	1.0	**.00**	**.00**	**.00**	**.00**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Yarowsky			1.0	1.0			**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
Yarowsky-cautious				1.0			**.04**	**.04**	**.04**	**.04**	**.04**	**.04**	**.00**	**.00**	**.00**	**.00**	**.04**	**.04**	**.04**	**.04**	**.04**	**.04**	**.04**	**.04**
Yarowsky-sum					1.0			1.0	1.0	1.0	1.0	1.0	**.00**	**.00**	**.00**	**.00**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Yarowsky-cautious-sum								1.0	1.0	1.0	1.0	1.0	**.00**	**.00**	**.00**	**.00**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
HS-bipartite avg-avg									1.0	1.0	1.0	1.0	**.00**	**.00**	**.00**	**.00**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
HS-bipartite avg-maj										1.0	1.0	1.0	**.00**	**.00**	**.00**	**.00**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
HS-bipartite maj-avg											1.0	1.0	**.00**	**.00**	**.00**	**.00**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
HS-bipartite maj-maj												1.0	**.00**	**.00**	**.00**	**.00**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
HS-bipartite maj-maj																	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Yar.-prop ϕ-θ																		1.0	1.0	1.0	1.0	1.0	1.0	1.0
Yar.-prop π-θ																			1.0	1.0	1.0	1.0	1.0	1.0
Yar.-prop θ-only																				1.0	1.0	1.0	1.0	1.0
Yar.-prop θ^{T}-only																					1.0	1.0	1.0	1.0
Yar.-prop-cautious ϕ-θ																						1.0	1.0	1.0
Yar.-prop-cautious π-θ																							1.0	1.0
Yar.-prop-cautious θ-only																								1.0
Yar.-prop-cautious θ^{T}-only																								

Table A.6: Statistical significance on the "land" WSD task, showing p values for differences in non-seeded accuracy (clean and noise accuracy are the same on this task). **Bold** values indicate differences which are significant at $p < 0.05$. † indicates differences which are not significant in non-seeded accuracy but are significant in full accuracy; all other signifances (but not necessarily p values) are the same for both accuracy measures.

	Seed DL	Single best label	DL-CoTrain (non-cautious)	DL-CoTrain (cautious)	Yarowsky	Yarowsky-cautious	Yarowsky-sum	Yarowsky-cautious-sum	HS-bipartite avg-avg	HS-bipartite avg-maj	HS-bipartite maj-avg	HS-bipartite maj-maj	EM	Hard EM	Online EM	Hard online EM	Yar.-prop ϕ-θ	Yar.-prop π-θ	Yar.-prop θ-only	Yar.-prop θ^{T}-only	Yar.-prop-cautious ϕ-θ	Yar.-prop-cautious π-θ	Yar.-prop-cautious θ-only	Yar.-prop-cautious θ^{T}-only
Seed DL		1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
Single best label	1.0		1.0†	**.00**	**.00**	**.00**	**.00**	**.00**	1.0†	.25†	.25†	.25†	**.00**	**.00**	.12†	.10†	.12†	1.0†	1.0†	1.0†	1.0†	**.00**	**.00**	**.00**	1.0†
DL-CoTrain (non-cautious)				1.0	**.00**	**.00**	**.00**	**.00**	1.0	.25	.25	.25	**.00**	**.00**	.12	.10	.12	1.0	1.0	1.0	1.0	**.00**	**.00**	**.00**	1.0
DL-CoTrain (cautious)					**.00**	**.00**	**.00**	**.00**	**.00**	.25	.25	.25	.88	.43	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	.13	**.00**	**.00**
Yarowsky						1.0	**.00**	1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	.69	.68	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
Yarowsky-cautious							1.0	**.00**	1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
Yarowsky-sum								1.0	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	.69	.68	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
Yarowsky-cautious-sum									**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
HS-bipartite avg-avg										1.0	.25	.25	**.00**	**.00**	.12	.10	.12	1.0	1.0	1.0	1.0	**.00**	**.00**	**.00**	1.0
HS-bipartite avg-maj											1.0	1.0	**.00**	**.00**	.18	.15	1.0	.25	.25	.25	.25	**.00**	**.00**	**.00**	.25
HS-bipartite maj-avg												1.0	**.00**	**.00**	.18	.15	1.0	.25	.25	.25	.25	**.00**	**.00**	**.00**	.25
HS-bipartite maj-maj													**.00**	**.00**	.18	.15	1.0	.25	.25	.25	.25	**.00**	**.00**	**.00**	.25
Yar.-prop ϕ-θ																		1.0	.12	.12	.12	**.00**	**.00**	**.00**	.12
Yar.-prop π-θ																			1.0	1.0	1.0	**.00**	**.00**	**.00**	1.0
Yar.-prop θ-only																				1.0	1.0	**.00**	**.00**	**.00**	1.0
Yar.-prop θ^{T}-only																					1.0	**.00**	**.00**	**.00**	1.0
Yar.-prop-cautious ϕ-θ																						1.0	.49	**.00**	
Yar.-prop-cautious π-θ																							1.0	**.00**	
Yar.-prop-cautious θ-only																								1.0	**.00**
Yar.-prop-cautious θ^{T}-only																									1.0

Table A.7: Statistical significance on the "sentence" WSD task, showing p values for differences in non-seeded accuracy (clean and noise accuracy are the same on this task). **Bold** values indicate differences which are significant at $p < 0.05$. † indicates differences which are not significant in non-seeded accuracy but are significant in full accuracy; all other signifances (but not necessarily p values) are the same for both accuracy measures.

Algorithm, iterations		Task							
		named entity		drug		land		sentence	
Yar.-prop ϕ-θ	1	80.39	77.89	53.63	51.80	78.36	77.72	55.34	51.88
	2	*78.11*	*75.32*	55.44	53.74	78.36	77.72	54.56	51.05
	3	78.68	75.96	*58.55*	*57.06*	78.36	77.72	54.56	51.05
	4	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
	5	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
Yar.-prop π-θ	1	78.34	75.58	54.15	52.35	78.36	77.72	54.56	51.05
	2	78.11	75.32	55.44	53.74	78.36	77.72	54.56	51.05
	3	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
	4	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
	5	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
Yar.-prop θ-only	1	78.56	75.84	54.66	52.91	78.36	77.72	54.56	51.05
	2	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
	3	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
	4	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
	5	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
Yar.-prop θ^{T}-only	1	77.88	75.06	52.07	50.14	78.36	77.72	54.56	51.05
	2	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
	3	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
	4	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
	5	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
Yar.-prop-cautious ϕ-θ	1	90.19	88.95	56.99	55.40	78.36	77.72	74.17	72.18
	2	89.17	87.79	58.55	57.06	78.36	77.72	*70.10*	*67.78*
	3	*78.91*	*76.22*	56.99	55.40	78.36	77.72	*54.56*	*51.05*
	4	*77.88*	*75.06*	57.51	55.96	78.36	77.72	*54.56*	*51.05*
	5	*45.84*	*45.89*	52.07	50.14	78.36	77.72	*54.56*	*51.05*
Yar.-prop-cautious π-θ	1	89.40	88.05	58.55	57.06	78.36	77.72	70.10	67.78
	2	89.17	87.79	58.55	57.06	78.36	77.72	70.10	67.78
	3	*77.88*	*75.06*	57.51	55.96	78.36	77.72	*54.56*	*51.05*
	4	*77.88*	*75.06*	57.51	55.96	78.36	77.72	*54.56*	*51.05*
	5	*45.84*	*45.89*	57.25	55.68	78.36	77.72	*54.56*	*51.05*
Yar.-prop-cautious θ-only	1	92.47	91.52	58.55	57.06	78.36	77.72	75.15	73.22
	2	*78.56*	*75.84*	57.51	55.96	78.36	77.72	*54.56*	*51.05*
	3	*45.84*	*45.89*	58.03	56.51	78.36	77.72	*54.56*	*51.05*
	4	*45.84*	*45.89*	52.07	50.14	78.36	77.72	*54.56*	*51.05*
	5	*32.95*	*33.42*	52.07	50.14	78.36	77.72	*54.56*	*51.05*
Yar.-prop-cautious θ^{T}-only	1	78.45	75.71	58.29	56.79	78.36	77.72	54.56	51.05
	2	*45.84*	*45.89*	58.03	56.51	78.36	77.72	54.56	51.05
	3	*45.84*	*45.89*	52.07	50.14	78.36	77.72	54.56	51.05
	4	*32.95*	*33.42*	52.07	50.14	78.36	77.72	54.56	51.05
	5	*32.95*	*33.42*	52.07	50.14	78.36	77.72	54.56	51.05

Table A.8: Percent clean accuracy (full and non-seeded respectively in each box) for Yarowsky-prop with different numbers of iterations of the graph propagation updates per Yarowsky-prop iteration. *Italic* values indicate differences which are significant versus a single iteration.

Algorithm, iterations		Task							
		named entity		drug		land		sentence	
Yar.-prop ϕ-θ	1	73.28	70.22	53.63	51.80	78.36	77.72	55.34	51.88
	2	*71.21*	*67.90*	55.44	53.74	78.36	77.72	54.56	51.05
	3	71.73	68.48	*58.55*	*57.06*	78.36	77.72	54.56	51.05
	4	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
	5	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
Yar.-prop π-θ	1	71.41	68.13	54.15	52.35	78.36	77.72	54.56	51.05
	2	71.21	67.90	55.44	53.74	78.36	77.72	54.56	51.05
	3	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
	4	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
	5	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
Yar.-prop θ-only	1	71.62	68.37	54.66	52.91	78.36	77.72	54.56	51.05
	2	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
	3	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
	4	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
	5	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
Yar.-prop θ^{T}-only	1	71.00	67.67	52.07	50.14	78.36	77.72	54.56	51.05
	2	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
	3	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
	4	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
	5	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
Yar.-prop-cautious ϕ-θ	1	82.22	80.19	56.99	55.40	78.36	77.72	74.17	72.18
	2	81.29	79.14	58.55	57.06	78.36	77.72	*70.10*	*67.78*
	3	*71.93*	*68.71*	56.99	55.40	78.36	77.72	*54.56*	*51.05*
	4	*71.00*	*67.67*	57.51	55.96	78.36	77.72	*54.56*	*51.05*
	5	*41.79*	*41.37*	52.07	50.14	78.36	77.72	*54.56*	*51.05*
Yar.-prop-cautious π-θ	1	81.50	79.37	58.55	57.06	78.36	77.72	70.10	67.78
	2	81.29	79.14	58.55	57.06	78.36	77.72	70.10	67.78
	3	*71.00*	*67.67*	57.51	55.96	78.36	77.72	*54.56*	*51.05*
	4	*71.00*	*67.67*	57.51	55.96	78.36	77.72	*54.56*	*51.05*
	5	*41.79*	*41.37*	57.25	55.68	78.36	77.72	*54.56*	*51.05*
Yar.-prop-cautious θ-only	1	84.30	82.50	58.55	57.06	78.36	77.72	75.15	73.22
	2	*71.62*	*68.37*	57.51	55.96	78.36	77.72	*54.56*	*51.05*
	3	*41.79*	*41.37*	58.03	56.51	78.36	77.72	*54.56*	*51.05*
	4	*41.79*	*41.37*	52.07	50.14	78.36	77.72	*54.56*	*51.05*
	5	*30.04*	*30.13*	52.07	50.14	78.36	77.72	*54.56*	*51.05*
Yar.-prop-cautious θ^{T}-only	1	71.52	68.25	58.29	56.79	78.36	77.72	54.56	51.05
	2	*41.79*	*41.37*	58.03	56.51	78.36	77.72	54.56	51.05
	3	*41.79*	*41.37*	52.07	50.14	78.36	77.72	54.56	51.05
	4	*30.04*	*30.13*	52.07	50.14	78.36	77.72	54.56	51.05
	5	*30.04*	*30.13*	52.07	50.14	78.36	77.72	54.56	51.05

Table A.9: Percent noise accuracy (full and non-seeded respectively in each box) for Yarowsky-prop with different numbers of iterations of the graph propagation updates per Yarowsky-prop iteration. *Italic* values indicate differences which are significant versus a single iteration.

Appendix B

Choosing seed rules

In our main experiments we simply used seed rules reported in earlier work. Now we consider ways to generate good seed rules and evaluate some simple choices on the named entity classification task. This appendix provides an initial look at choosing seed rules, and also provides an indication of whether or not Collins and Singer (1999)'s results could be easily replicated without having their seed rules available. Outside of this chapter we use Collins and Singer (1999)'s seed rules for all experiments on the named entity classification task.

Some simple general methods for choosing seeds with assisted manual selection include:

1. Choose features which occur with high frequency in the training data and which appear in manual evaluation to have a clear associated label.

2. Use method 1, but additionally prefer features which occur in a variety of contexts.

Eisner and Karakos (2005) and Zagibalov and Carroll (2008) discuss more sophisticated ways to choose good seeds automatically or with little manual effort, which we do not evaluate here.

We experimented with three different seed choice methods where we sorted the features from the training data by some criterion, and then manually extracted good (in our personal judgement) features from near the top of the list. Features indicative of locations tend to appear infrequently near the top of the ordered list with these methods, and some good location seeds may have been overlooked. The sorting methods are:

Frequency Sorting by frequency of occurrence.

Contexts Sorting by the number of other features that each feature occurs with.

Weighted Sorting by the weighted count of the other features that each feature occurs with. Weights are the overall frequency of occurrence as a fraction of the total number of features.

With each method we chose fifteen seed rules, and then made smaller seed rule lists by taking the top three, six, nine, and twelve rules, evenly split between the three labels. These seeds are shown in fig. B.1.

To compare these seed rules against Collins and Singer (1999)'s results we used their algorithms on the named entity classification task. Since our seed rules use both spelling and context features (whereas Collins and Singer (1999)'s seeds use only spelling features), we take advantage of our slightly generalized form of DL-CoTrain described in section 4.1 to support seeds from both views. We gave the seed rules a weight of 0.9999 when adding them to subsequently learned decision lists. Table B.1 shows the final test accuracies. No method we consider here can consistently perform as well as Collins and Singer (1999)'s seed rules without using more rules. However, the frequency and contexts methods are statistically equivalent to Collins and Singer (1999)'s seeds at nine rules with Yarowsky-cautious and at 6 rules with DL-CoTrain cautious. With Yarowsky non-cautious the frequency method requires 12 rules to become equivalent and the contexts method requires 15. The weighted method generally does not perform as well as the others, especially with the Yarowsky and Yarowsky-cautious algorithms.

While producing these seed rules, we noticed that features indicative of some labels appear in different quantities in the top ranked features of different orderings. For example, location features were rare near the top of our orderings, but may be more common in other orderings. Thus we believe that with a wider selection of sorting options it might be worthwhile to manually select the seeds for each label separately from differently ordered lists. Similarly, it may not be necessary or desirable to use the same number of seed rules for each label, as we see that Collins and Singer (1999)'s seed rules work well with only one rule for the person label.

Feature	Label
X0_U.S.	location
X01_president	person
X2_Incorporated	organization
X0_Japan	location
X2_Mr.	person
X2_Corporation	organization
X0_California	location
X01_chairman	person
X11_stake-in	organization
X0_New-York	location
X01_analyst	person
X2_Group	organization
X11_court-in	location
X01_director	person
X2_Bank	organization

(a) Frequency

Feature	Label
X0_U.S.	location
X01_president	person
X2_Incorporated	organization
X11_plant-in	location
X2_Mr.	person
X2_Corporation	organization
X0_Japan	location
X01_chairman	person
X01_unit	organization
X0_California	location
X01_director	person
X11_unit-of	organization
X11_court-in	location
X01_analyst	person
X11_stake-in	organization

(b) Contexts

Feature	Label
X0_N.J.	location
X2_Mr.	person
X2_Incorporated	organization
X0_N.Y.	location
X2_James	person
X2_Corporation	organization
X0_U.S.	location
X2_John	person
X2_Group	organization
X0_California	location
X2_Smith	person
X2_Limited	organization
X0_Ohio	location
X2_Robert	person
X2_International	organization

(c) Weighted

Figure B.1: Different choices of seed rules for the named entity classification task. Each list shows all 15 rules that we selected. When using fewer rules we took the first 3, 6, 9, or 12 rules from the top. Compare to Collins and Singer (1999)'s seed rules in fig. 2.3.

Seed rules		Algorithm					
Choice	Frequency	DL-CoTrain	(cautious)	Yarowsky		Yarowsky-cautious	
C&S	7	91.56	90.49	83.58	81.49	91.11	89.97
Frequency	3	*53.93*	*47.93*	*76.51*	*73.51*	*84.38*	*82.43*
	6	90.65	88.54	*78.56*	*73.55*	*84.49*	*80.91*
	9	91.22	89.02	*80.50*	*75.07*	91.45	89.32
	12	91.68	89.13	85.06	80.12	91.79	89.29
	15	91.68	88.80	*85.52*	80.16	91.56	88.64
Contexts	3	*53.93*	*47.93*	*76.51*	*73.51*	*84.38*	*82.43*
	6	90.31	88.19	*77.31*	*72.15*	*83.35*	*79.61*
	9	90.65	88.35	*78.68*	*72.86*	89.62	87.02
	12	91.79	89.51	*80.73*	*74.54*	91.79	89.51
	15	91.68	88.91	84.61	78.94	90.99	87.94
Weighted	3	*88.26*	*87.17*	*71.95*	*69.36*	*87.57*	*86.43*
	6	*88.03*	*86.17*	*74.34*	*70.36*	*87.80*	*85.90*
	9	90.99	89.13	*75.26*	*70.15*	*82.10*	*78.40*
	12	90.88	88.89	*76.62*	*71.31*	*82.90*	*79.04*
	15	91.56	89.54	*77.65*	*72.06*	*84.49*	*80.66*

(a) Clean accuracy

Seed rules		Algorithm					
Choice	Frequency	DL-CoTrain	(cautious)	Yarowsky		Yarowsky-cautious	
C&S	7	83.47	81.58	76.20	73.46	83.06	81.11
Frequency	3	*49.17*	*43.19*	*69.75*	*66.24*	*76.92*	*74.27*
	6	82.64	79.04	*71.62*	*65.66*	*77.03*	*72.22*
	9	83.16	79.05	*73.39*	*66.67*	83.37	79.31
	12	83.58	78.74	77.55	70.78	83.68	78.88
	15	83.58	78.17	*77.96*	70.56	83.47	78.03
Contexts	3	*49.17*	*43.19*	*69.75*	*66.24*	*76.92*	*74.27*
	6	82.33	78.77	*70.48*	*64.45*	*75.99*	*71.11*
	9	82.64	78.51	*71.73*	*64.74*	81.70	77.33
	12	83.68	79.13	*73.60*	*65.89*	83.68	79.13
	15	83.58	78.22	77.13	69.45	82.95	77.37
Weighted	3	*80.46*	*78.83*	*65.59*	*62.73*	*79.83*	*78.15*
	6	*80.25*	*77.49*	*67.78*	*63.27*	*80.04*	*77.25*
	9	82.95	79.80	*68.61*	*62.81*	*74.84*	*70.20*
	12	82.85	79.40	*69.85*	*63.69*	*75.57*	*70.60*
	15	83.47	79.82	*70.79*	*64.24*	*77.03*	*71.90*

(b) Noise accuracy

Table B.1: Percent accuracy (full and non-seeded respectively in each box) with different methods of seed selection. The "C&S" row is our results with the seeds of Collins and Singer (1999) (fig. 2.3, and see table A.1 for their own results with these rules). *Italics* indicate a statistically significant difference versus C&S.

Appendix C

Implementing Collins and Singer's algorithms

In this chapter we discuss additional implementation details for the decision list-based algorithms of Collins and Singer (1999), which may be useful for replicating experiments.

C.1 Decision list tie breaking

When learning a decision list, including inside the Yarowsky algorithm and the other decision list-based algorithms, the scoring function used to order rules sometimes produces ties. Similarly when applying thresholding and cautiousness the ordering used for selecting rules can produce ties. This introduces a sensitivity to how such ties are broken. In this appendix we evaluate this sensitivity and provide details which may be useful for replicating Collins and Singer (1999)'s results or our own.

Any sensitivity to tie breaking is likely exaggerated by cautiousness, where early during bootstrapping there are only a few rules and tie breaking will effect which rules fall under the cutoff. Similar sensitivity exists even without cautiousness not only because the threshold also cuts off some rules, but also because tie breaking may effect early labellings which will in turn effect the rest of training. In the case of the named entity task this sensitivity may also be exaggerated by the ambiguities described in appendix D.5; similar minor data processing errors are likely to occur in other data sets as well.

Collins and Singer (1999) do not give any details as to how they resolved ties in basic

decision list learning or in cautiousness. We break both kinds of ties by using fixed orders for labels and features. A tie is broken first by the label order and then (if the labels are the same) by the feature order.

In evaluating the behaviour and sensitivity of our decision list implementation we focused on Collins and Singer (1999)'s algorithms on the named entity task, since we were concerned primarily with ensuring that the details of our decision list learning match Collins and Singer (1999)'s as closely as possible. We tried all six orders of the three labels (named entity classes) and we tried three simple methods for choosing a feature order:

Lexicographic Lexicographic order of the features using the feature names as in the data files (as shown in appendix D) including type prefixes.

Reverse lexicographic Reverse lexicographic order of the features.

Order seen The order in which features are first seen in the training data.

We also tried random feature orders to evaluate sensitivity.

Tables C.1 and C.2 show test accuracy results as the label and feature orders are changed independently. There is some variation, and in these experiments no choice gives exactly the same results as Collins and Singer (1999) (see table A.1).[1] However, we do not find any statistically significant differences. In our experiments outside this appendix we use the location-person-organization label ordering (following the label numbering of 1, 2, 3 used in the data files) and the lexicographic feature ordering.

C.2 Smoothed and unsmoothed thresholding

Collins and Singer (1999) describe checking the threshold ζ against unsmoothed decision list scores for DL-CoTrain cautious and against the smoothed scores (3.2) for Yarowsky non-cautious (see sections 3.2 and 3.3). They motivate using unsmoothed thresholding for DL-CoTrain cautious by noting that cautiousness in taking only the top n rules by frequency handles low-frequency rules on its own. We take this to indicate that in general non-cautious algorithms should use smoothed thresholding and cautious algorithms should use unsmoothed thresholding. In earlier results (including those reported in Whitney and Sarkar (2012)) we incorrectly used unsmoothed thresholding for non-cautious algorithms, and in correcting

[1] But we did not test the various other pairings of label order and feature order.

Order	Algorithm					
	DL-CoTrain (cautious)		Yarowsky		Yarowsky-cautious	
Loc.-per.-org.	91.56	90.49	83.58	81.49	91.11	89.97
Loc.-org.-per.	91.68	90.62	83.24	81.11	91.22	90.10
Per.-loc.-org.	91.56	90.49	83.69	81.62	90.99	89.85
Per.-org.-loc.	91.56	90.49	83.58	81.49	90.99	89.85
Org.-loc.-per.	91.68	90.62	83.12	80.98	91.22	90.10
Org.-per.-loc.	91.68	90.62	83.24	81.11	91.11	89.97
Mean	91.62	90.56	83.41	81.30	91.11	89.97
±	0.06	0.07	0.22	0.24	0.09	0.10

(a) Clean accuracy

Order	Algorithm					
	DL-CoTrain (cautious)		Yarowsky		Yarowsky-cautious	
Loc.-per.-org.	83.47	81.58	76.20	73.46	83.06	81.11
Loc.-org.-per.	83.58	81.69	75.88	73.12	83.16	81.23
Per.-loc.-org.	83.47	81.58	76.30	73.58	82.95	81.00
Per.-org.-loc.	83.47	81.58	76.20	73.46	82.95	81.00
Org.-loc.-per.	83.58	81.69	75.78	73.00	83.16	81.23
Org.-per.-loc.	83.58	81.69	75.88	73.12	83.06	81.11
Mean	83.52	81.63	76.04	73.29	83.06	81.11
±	0.05	0.05	0.20	0.22	0.09	0.09

(b) Noise accuracy

Table C.1: Percent accuracy (full and non-seeded respectively in each box) with different label orders for tie breaking, using the lexicographic feature order. "Mean" is the mean and percent standard deviation over all orders. No result has a statistically significant difference versus the location-person-organization order.

Order	Algorithm					
	DL-CoTrain (cautious)		Yarowsky		Yarowsky-cautious	
Lexicographic	91.56	90.49	83.58	81.49	91.11	89.97
Reverse lexicographic	90.76	89.59	83.58	81.49	90.65	89.46
Order seen	90.76	89.59	83.58	81.49	90.88	89.72
Random	91.07	89.94	83.58	81.49	90.84	89.68
$\pm$	0.24	0.27	0.00	0.00	0.25	0.29

(a) Clean accuracy

Order	Algorithm					
	DL-CoTrain (cautious)		Yarowsky		Yarowsky-cautious	
Lexicographic	83.47	81.58	76.20	73.46	83.06	81.11
Reverse lexicographic	82.74	80.76	76.20	73.46	82.64	80.65
Order seen	82.74	80.76	76.20	73.46	82.85	80.88
Random	83.02	81.08	76.20	73.46	82.82	80.84
$\pm$	0.22	0.24	0.00	0.00	0.23	0.26

(b) Noise accuracy

Table C.2: Percent accuracy (full and non-seeded respectively in each box) with different feature orders for tie breaking, using the location-person-organization label order. "Random" is a mean and percent standard deviation over 10 orders chosen uniformly at random. No result, including the individual random orders (not shown here) has a statistically significant difference versus the lexicographic order.

them we noticed some differences in results. In this appendix we evaluate the effects of using smoothed and unsmoothed scores for thresholding.

Table C.3 shows the test accuracies for Yarowsky, DL-CoTrain cautious, Yarowsky-cautious, and the sum variants for Yarowsky and Yarowsky-cautious, with both smoothed and unsmoothed thresholding. The smoothed scores are always used to order the decision lists regardless of how thresholding is done. Outside of this appendix we always use smoothed thresholding for non-cautious forms of Collins and Singer (1999)'s algorithms and unsmoothed thresholding for cautious forms of the same.

On the named entity task the results agree with Collins and Singer (1999)'s choice, with the standard thresholding for each algorithm performing significantly better than the alternative. On the "drug" task the standard thresholding (smoothed) is again better for Yarowsky non-cautious, Yarowsky-sum non-cautious, and DL-CoTrain cautious (although the difference is not statistically significant for Yarowsky non-cautious), but the thresholding makes no difference to the Yarowsky-cautious and Yarowsky-cautious-sum results. Recall that this task has an especially small training set and a much larger test set (see section 2.2), so that it may be particularly sensitive to the handling of infrequent rules. On the "land" task we see little change, as expected since most algorithms find the same accuracy on this task (see section 2.2 and other results). On the "sentence" task the standard thresholding (unsmoothed) for Yarowsky-cautious, Yarowsky-cautious-sum, and DL-CoTrain cautious performs significantly better than the alternative, but Yarowsky and Yarowsky-sum perform significantly better with unsmoothed thresholding than with the standard smoothed thresholding.

Thresholding	Algorithm	Task							
		named entity		drug		land		sentence	
Unsmoothed	Yarowsky	*81.19*	*78.79*	55.70	54.02	79.03	78.41	*62.91*	*60.04*
	Yar.-sum	*82.33*	*80.08*	*53.63*	*51.80*	78.16	77.51	*65.44*	*62.76*
	DL-CoTrain cautious†	***91.56***	***90.49***	*59.59*	*58.17*	78.36	77.72	*68.16*	*65.69*
	Yar.-cautious†	*91.11*	*89.97*	54.40	52.63	79.10	78.48	***78.64***	***76.99***
	Yar.-cautious-sum†	***91.56***	***90.49***	54.40	52.63	78.36	77.72	***78.64***	***76.99***
Smoothed	Yarowsky†	*83.58*	*81.49*	59.07	57.62	79.03	78.41	*58.06*	*54.81*
	Yar.-sum†	*85.06*	*83.16*	***60.36***	***59.00***	78.36	77.72	*58.06*	*54.81*
	DL-CoTrain cautious	*89.17*	*87.79*	*54.15*	*52.35*	78.36	77.72	*54.56*	*51.05*
	Yar.-cautious	*89.51*	*88.17*	54.40	52.63	**79.17**	**78.55**	*66.60*	*64.02*
	Yar.-cautious-sum	*90.31*	*89.07*	54.40	52.63	78.36	77.72	*66.60*	*64.02*

(a) Clean accuracy

Thresholding	Algorithm	Task							
		named entity		drug		land		sentence	
Unsmoothed	Yarowsky	*74.01*	*71.03*	55.70	54.02	79.03	78.41	*62.91*	*60.04*
	Yar.-sum	*75.05*	*72.19*	*53.63*	*51.80*	78.16	77.51	*65.44*	*62.76*
	DL-CoTrain cautious†	***83.47***	***81.58***	*59.59*	*58.17*	78.36	77.72	*68.16*	*65.69*
	Yar.-cautious†	*83.06*	*81.11*	54.40	52.63	79.10	78.48	***78.64***	***76.99***
	Yar.-cautious-sum†	***83.47***	***81.58***	54.40	52.63	78.36	77.72	***78.64***	***76.99***
Smoothed	Yarowsky†	*76.20*	*73.46*	59.07	57.62	79.03	78.41	*58.06*	*54.81*
	Yar.-sum†	*77.55*	*74.97*	***60.36***	***59.00***	78.36	77.72	*58.06*	*54.81*
	DL-CoTrain cautious	*81.29*	*79.14*	*54.15*	*52.35*	78.36	77.72	*54.56*	*51.05*
	Yar.-cautious	*81.60*	*79.49*	54.40	52.63	**79.17**	**78.55**	*66.60*	*64.02*
	Yar.-cautious-sum	*82.33*	*80.30*	54.40	52.63	78.36	77.72	*66.60*	*64.02*

(b) Noise accuracy

Table C.3: Percent accuracy (full and non-seeded respectively in each box) for Collins and Singer (1999)'s Yarowsky, DL-CoTrain cautious, and Yarowsky-cautious algorithms and our sum variants, with the threshold checked against smoothed and unsmoothed scores. **Bold** indicates a maximum in a column. *Italics* indicate results which are statistically significant versus the corresponding results with the other method of thresholding. † indicates the standard forms that we use for experiments outside this appendix.

Appendix D

Using the named entity classification data

Section 2.1 described the named entity classification task from Collins and Singer (1999), and the corresponding data which was provided to us by Michael Collins. In this appendix we describe in more detail issues and properties of the data that we noticed while using it for experiments. These details may be necessary for replicating Collins and Singer (1999)'s results or our own.

By the readme file included with the data, the features with the prefixes X01_, X11_, and X3_ are context features and all others are spelling features. The readme does not give meanings for the different feature prefixes, but table D.1 shows the apparent meanings for each type that occurs in the data.

D.1 Temporal expressions

Collins and Singer (1999) describe filtering the test data to remove temporal expressions, which they imply to be (week) day and month names. We define a temporal expression to be any example which contains one of the strings shown in table D.2 in any feature. There are 38 such examples in the test data, which matches Collins and Singer (1999)'s number exactly. There are 4024 such examples in the training data. We remove temporal expressions from the test data as Collins and Singer (1999) describe, but not from the training data; see appendix D.7.

Feature type prefix	Name in paper	Meaning
X0_	full-string=x	Full phrase to be classified.
X2_	contains(x)	Individual word in the phrase to be classified.
X5_	allcap1	Flag set if X0_ is a single word in all capital letters.
X6_	allcap2	Flag set if X0_ is a single word in capital letters and periods.
X7_	nonalpha=x	Contains any non-alphabetical characters from X0_
X42_	?	Seemingly a decomposition of X11_ features; see appendix D.2.
X01_	context=x	Appositive modifier context. The head of the appositive.
X11_	context=x	Prepositional context. The preposition and the noun it modifies.
X3_	context-type=x	Type of context. "LEFT" for an appositive modifier and "RIGHT" for a preposition.

Table D.1: Feature types by prefix, with apparent meanings and corresponding names in the text of Collins and Singer (1999). "?" indicates that the feature does not appear to be mentioned in the paper.

January	June	November	Thursday
February	July	December	Friday
March	August	Monday	Saturday
April	September	Tuesday	Sunday
May	October	Wednesday	

Table D.2: Strings representing temporal expressions, which we used to filter Collins and Singer (1999)'s named entity classification test data.

D.2 Features only present in the test data

There are 321 features which occur in the test data but not in the training data. These are all prefixed with X42_, and appear to be special decompositions of hyphenated X11_ features which represent context in the case that the noun phrase to be classified is a complement to a preposition. For example, the following appears in the test data:

X0_Steptoe X11_partner-at X42_partner X42_at X3_RIGHT

There are 569 test examples which contain such features.

Following the readme file, we consider X42_ features to be spelling features (such as for co-training views) even though they appear to be decompositions of context features. However, since the X42_ features are not in the training data they do not effect results.

D.3 Empty features

There is one example in the training data with empty-valued features. It looks like this:

X0_ X01_ X5_ALLCAP X3_LEFT

There are no empty-valued features in the test data.

D.4 Duplicated examples

Considering examples with exactly the same features to be identical, in the training data there are 8166 distinct examples which occur more than once, and 70512 unique examples in total. In the test data there are 14 distinct examples which occur more than once, and 948 unique examples in total. There are 418 distinct examples which occur in both the training data and the test data. There are 1125 instances of such examples in the training data and 425 in the test data.

D.5 Ambiguous examples and errors

Ten of the examples in the training data contain both X2_Mr. and X2_Incorporated features, and therefore are ambiguous with respect to the seed rules (see fig. 2.3). These examples are shown in fig. D.1. There are no such examples in the test data.

- X3_RIGHT X2_Brothers X7_. X2_Salomon X2_Mr. X2_Incorporated X0_Salomon-Brothers-Incorporated-Mr.-LipsNNP X2_LipsNNP X11_president-with
- X3_RIGHT X2_N.T.C. X11_subsidiary-of X2_Davis X7_.... X2_Mr. X2_Incorporated X2_Group X0_N.T.C.-Group-Incorporated-Mr.-Davis
- X3_RIGHT X2_FishNNP X7_. X11_officer-at X2_Mr. X2_Incorporated X0_Tele-Communications-Incorporated-Mr.-FishNNP X2_Tele-Communications
- X0_Hallmark-Cards-Incorporated-Mr.-WernerNNP X3_RIGHT X11_unit-of X2_Hallmark X7_. X2_Cards X2_Mr. X2_Incorporated X2_WernerNNP
- X0_C.R.T.-Services-Incorporated-Mr.-DonneNNP X3_RIGHT X2_C.R.T. X2_Mr. X7_..... X2_Services X11_broker-with X2_Incorporated X2_DonneNNP
- X3_RIGHT X11_unit-of X2_PepsiCo X7_. X2_Mr. X2_Incorporated X2_Jordan X0_PepsiCo-Incorporated-Mr.-Jordan
- X3_RIGHT X2_Lewis X11_president-of X7_. X2_Mr. X2_Toys X2_Galoob X2_Incorporated X0_Lewis-Galoob-Toys-Incorporated-Mr.-NNP X2_NNP
- X2_Mr. X0_Stoneridge-Resources-Incorporated-Mr.-NNP X3_RIGHT X7_. X2_Stoneridge X2_Resources X11_officer-of X2_Incorporated X2_NNP
- X3_RIGHT X2_Air X7_.... X11_subsidiary-of X2_Mr. X0_U.S.-Air-Group-Incorporated-Mr.-BurNNP X2_Incorporated X2_BurNNP X2_Group X2_U.S.
- X3_RIGHT X2_Industries X11_unit-of X2_Fairchild X0_Fairchild-Industries-Incorporated-Mr.-NNP X7_. X2_Mr. X2_Incorporated X2_NNP

Figure D.1: The examples in Collins and Singer (1999)'s named entity classification training data which are ambiguous with respect to the seed rules.

1. In essence , that would not support suppositions that the economy is slowing down remarkably , " said John Lipsky , an economist and vice president with Salomon Brothers Inc. Mr. Lipsky predicted the unemployment rate will probably remain unchanged at 5.4 % in October .

2. He 's a pretty hard competitor , " says Alan Davis , president of Bibb Co . , a textile subsidiary of NTC Group Inc. Mr. Davis knows Mr. Lanier well since they both emerged as victors in a bloody takeover battle for West Point 's archrival , J.P. Stevens & Co . , earlier this year that resulted in the carve-up of Stevens .

Figure D.2: Sentences in the Wall Street Journal corpus which appear to match the first two ambiguous examples (respectively) of fig. D.1.

In Collins and Singer (1999)'s algorithms (see chapter 3), the seed rules will all be assigned the same score and therefore their relative order is not specified. Having training examples where more than one seed rule applies is a potential problem since it introduces an unexpected sensitivity on which order the algorithm uses.

Moreover, the ten ambiguous examples appear to be errors in the data. Firstly, note that all but two of these examples contain "NNP" in features, presumably part of speech tags accidentally introduced during data processing. There are a total of 808 training and 12 test examples which contain the string "NNP" in any feature. Secondly, note that in each case the feature with prefix X0_, which represents the entire noun phrase to be classified, appears to be an organization name and personal name concatenated together.

Collins and Singer (1999) indicate that their data is from New York Times text. We did not manage to find good matches for named entity examples in the New York Times text that we have available, but we did find apparent matches in the Wall Street Journal corpus (WSJ88 as in Ratnaparkhi (1998)). Perhaps there is overlap between the two data sets due to shared newswire sources. The first two error examples seem to match the Wall Street Journal sentences shown in fig. D.2. Ignoring the mismatch in data source and the use of "Inc." rather than "Incorporated", these sentences suggest a further reason for the errors: both in fact contain two sentences which should have been broken apart after "Inc.". We can see the same pattern in the X0_ spelling features containing both an organization name and a personal name; the organization name always ends with "Incorporated" while the personal name always begins with "Mr.". Each sentence in fig. D.2 also contains an end-quote which is not matched by an open-quote in the sentence.

Seed rule precedence	Algorithm					
	DL-CoTrain (cautious)		Yarowsky		Yarowsky-cautious	
X2_Incorporated	91.56	90.49	83.58	81.49	91.11	89.97
X2_Mr.	91.56	90.49	83.81	81.75	90.76	89.59

(a) Clean accuracy

Seed rule precedence	Algorithm					
	DL-CoTrain (cautious)		Yarowsky		Yarowsky-cautious	
X2_Incorporated	83.47	81.58	76.20	73.46	83.06	81.11
X2_Mr.	83.47	81.58	76.40	73.70	82.74	80.76

(b) Noise accuracy

Table D.3: Percent accuracy (full and non-seeded respectively in each box) with both precedences for the ambiguous named entity classification seed rules, with no filtering on the training data (see appendix D.7). No differences between precedences are statistically significant.

D.6 Seed rule order

Due to the ambiguities discussed in appendix D.5, the relative precedence of the X2_Incorporated and X2_Mr. seed rules may change results. Since the seed rules are used as a decision list in these algorithms we can control the precedence simply by changing the order of the decision list.

To attempt to match the behaviour of Collins and Singer (1999) we evaluated our implementation of their algorithms on both distinct orders. Test accuracy results are shown in table D.3. The accuracies differ between the orders for Yarowsky and Yarowsky-cautious, but not for DL-CoTrain. For Yarowsky and Yarowsky-cautious there is no statistically significant difference, but giving the X2_Incorporated seed rule precedence produces results which are numerically closer to Collins and Singer (1999)'s reported results (see table A.1). Therefore we give the X2_Incorporated rule precedence in the experiments outside of this appendix.

D.7 Data set sizes and filtering

The data in the files provided to us includes a file of 1000 labelled test examples and a file of 89305 unlabelled examples which we take to be the training data. There are 123 noise

test examples (examples which are not in one of the three labels a classifier should learn). Collins and Singer (1999) describe filtering the test data to remove temporal expressions. Removing the temporal expression test examples described in appendix D.1 leaves 962 test examples, which is exactly the number reported in the paper. We therefore use the filtered test data for experiments. There are 85 noise test examples after filtering.

The paper reports using 87962 training examples (88962 examples including the 1000 test examples, which were presumably removed). This number does not match the size of the data file we take to be for training. This firstly raises the possibility that the file of 89305 examples is in fact the combined data before removing the test set; however, this cannot be the case since it does not contain all the test examples (see appendix D.4). Secondly, there is the possibility that the training data should be filtered, such as to remove temporal items or to address some of the issues that we note above. We considered several filtering methods:

- Remove all examples which are also present (with the same features) in the test data.
- Remove the first occurrence of each example that is present (with the same features) in the test data.
- Remove temporal expressions as for the test data.
- Remove examples containing the string "NNP" in any feature.
- Remove the items which are ambiguous with respect to the seed rules.

Table D.4 shows the data set sizes produced by these filtering methods. None matches the number of training examples reported by Collins and Singer (1999), nor does any obvious combination. We also compared the empirical results reported by Collins and Singer (1999) against our own results from their algorithms trained on the data filtered by each method. The test accuracies are shown in table D.5. Again no filtering method matches their results (see table A.1) exactly. There are few significant differences versus no filtering. We used no training set filtering (and therefore all 89305 training examples) for the experiments outside this appendix.

Filtering	# removed	# remaining
None	0	89305
All test examples	1125	88180
First occurrence test examples	418	88887
Temporal expressions	4024	85281
NNP examples	808	88497
Ambiguities	10	89295
Collins and Singer (1999) combined data	343	88962
Collins and Singer (1999) without test	1343	87962

Table D.4: Sizes of the named entity training data set with various filtering methods. The last two rows are the numbers reported by Collins and Singer (1999) for their complete data and for the complete data without the test set, respectively. The numbers of removed examples are relative to the 89305 examples in the data file.

Filtering	Algorithm					
	DL-CoTrain (cautious)		Yarowsky		Yarowsky-cautious	
None	91.56	90.49	83.58	81.49	91.11	89.97
All test examples	91.68	90.62	*84.83*	*82.90*	*89.62*	*88.30*
First occurrence test examples	91.68	90.62	83.69	81.62	91.33	90.23
Temporal expressions	90.76	89.59	*82.67*	*80.46*	90.54	89.33
NNP examples	91.56	90.49	83.81	81.75	91.56	90.49
Ambiguities	91.45	90.36	83.69	81.62	91.11	89.97

(a) Clean accuracy

Filtering	Algorithm					
	DL-CoTrain (cautious)		Yarowsky		Yarowsky-cautious	
None	83.47	81.58	76.20	73.46	83.06	81.11
All test examples	83.58	81.69	*77.34*	*74.74*	*81.70*	*79.61*
First occurrence test examples	83.58	81.69	76.30	73.58	83.26	81.34
Temporal expressions	82.74	80.76	*75.36*	*72.54*	82.54	80.53
NNP examples	83.47	81.58	76.40	73.70	83.47	81.58
Ambiguities	83.37	81.46	76.30	73.58	83.06	81.11

(b) Noise accuracy

Table D.5: Percent accuracy (full and non-seeded respectively in each box) with filtering methods, giving the X2_Incorporated seed precedence over the X2_Mr. seed rule (see appendix D.6). *Italics* indicate a statistically significant difference versus no filtering.

Bibliography

Abney, S. (2002). Bootstrapping. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 360–367, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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

Agichtein, E. and Gravano, L. (2000). Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*, DL '00.

Bengio, Y., Delalleau, O., and Le Roux, N. (2006). Label propagation and quadratic criterion. In Chapelle, O., Schölkopf, B., and Zien, A., editors, *Semi-Supervised Learning*, pages 193–216. MIT Press.

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

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of Computational Learning Theory*.

Castelli, V. and Cover, T. M. (1995). On the exponential value of labeled samples. *Pattern Recognition Letters*, 16(1):105–111.

Collins, M. (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. Association for Computational Linguistics.

Collins, M. and Singer, Y. (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.

Dasgupta, S., McAllester, D., and Littman, M. (2001). PAC generalization bounds for co-training. In *Proceedings of Neural Information Proccessing Systems*.

Daume, H. (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*.

Dempster, A. P., Laird, N. M., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1(39):1–38.

Eisner, J. and Karakos, D. (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. Association for Computational Linguistics.

Gale, W. A., Church, K. W., and Yarowsky, D. (1992). Using bilingual materials to develop word sense disambiguation methods. In *Proceedings of the 4th International Conference on Theoretical and Methodological Issues in Machine Translation*.

Haffari, G. and Sarkar, A. (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.

Haghighi, A. and Klein, D. (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. Association for Computational Linguistics.

Haghighi, A. and Klein, D. (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. Association for Computational Linguistics.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational Linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA. Association for Computational Linguistics.

ISI Hansards (2001). Aligned Hansards of the 36th Parliament of Canada. From the website of the Natural Language Group, USC Information Sciences Institute: http://www.isi.edu/natural-language/download/hansard.

Jiang, J. (2007). A literature survey on domain adaptation of statistical classifiers. http://sifaka.cs.uiuc.edu/jiang4/domain_adaptation/survey/da_survey.pdf.

Johnson, M. (2007). Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 296–305, Prague, Czech Republic. Association for Computational Linguistics.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (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.

Merialdo, B. (1994). Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–172.

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

Ratnaparkhi, A. (1998). Statistical models for unsupervised prepositional phrase attachment. In *Proceedings of the 17th international conference on Computational linguistics - Volume 2*, COLING '98, pages 1079–1085, Montreal, Quebec, Canada. Association for Computational Linguistics.

Riloff, E. and Shepherd, J. (1997). A corpus-based approach for building semantic lexicons. In *In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124.

Rivest, R. L. (1987). Learning decision lists. In *Machine Learning*, pages 229–246.

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

Subramanya, A., Petrov, S., and Pereira, F. (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. Association for Computational Linguistics.

Talukdar, P. P. (2010). *Graph-based weakly-supervised methods for information extraction & integration.* PhD thesis, University of Pennsylvania. Software: *https://github.com/parthatalukdar/junto.*

Whitney, M. and Sarkar, A. (2012). Bootstrapping via graph propagation. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 620–628, Jeju Island, Korea. Association for Computational Linguistics.

Yarowsky, D. (1994). Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 88–95, Las Cruces, New Mexico, USA. Association for Computational Linguistics.

Yarowsky, D. (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. Association for Computational Linguistics.

Zagibalov, T. and Carroll, J. (2008). Automatic seed word selection for unsupervised sentiment classification of Chinese text. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 1073–1080, Manchester, UK. Coling 2008 Organizing Committee.

Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

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

Zhu, X. and Goldberg, A. B. (2009). *Introduction to Semi-Supervised Learning*. Morgan & Claypool.