Understanding the Yarowsky Algorithm

Steven Abney*
University of Michigan

Many problems in computational linguistics are well suited for bootstrapping (semisupervised learning) techniques. The Yarowsky algorithm is a well-known bootstrapping algorithm, but it is not mathematically well understood. This article analyzes it as optimizing an objective function. More specifically, a number of variants of the Yarowsky algorithm (though not the original algorithm itself) are shown to optimize either likelihood or a closely related objective function K.

1. Introduction

Bootstrapping, or semisupervised learning, has become an important topic in computational linguistics. For many language-processing tasks, there are an abundance of unlabeled data, but labeled data are lacking and too expensive to create in large quantities, making bootstrapping techniques desirable.

The Yarowsky (1995) algorithm was one of the first bootstrapping algorithms to become widely known in computational linguistics. In brief, it consists of two loops. The "inner loop" or **base learner** is a supervised learning algorithm. Specifically, Yarowsky uses a simple decision list learner that considers rules of the form "If instance x contains feature f, then predict label j" and selects those rules whose precision on the training data is highest.

The "outer loop" is given a seed set of rules to start with. In each iteration, it uses the current set of rules to assign labels to unlabeled data. It selects those instances regarding which the base learner's predictions are most confident and constructs a labeled training set from them. It then calls the inner loop to construct a new classifier (that is, a new set of rules), and the cycle repeats.

An alternative algorithm, co-training (Blum and Mitchell 1998), has subsequently become more popular, perhaps in part because it has proven amenable to theoretical analysis (Dasgupta, Littman, and McAllester 2001), in contrast to the Yarowsky algorithm, which is as yet mathematically poorly understood. The current article aims to rectify this lack of understanding, increasing the attractiveness of the Yarowsky algorithm as an alternative to co-training. The Yarowsky algorithm does have the advantage of placing less of a restriction on the data sets it can be applied to. Co-training requires data attributes to be separable into two **views** that are conditionally independent given the target label; the Yarowsky algorithm makes no such assumption about its data.

In previous work, I did propose an assumption about the data called **precision independence**, under which the Yarowsky algorithm could be shown effective (Abney 2002). That assumption is ultimately unsatisfactory, however, not only because it

* 4080 Frieze Bldg., 105 S. State Street, Ann Arbor, MI 48109-1285. E-mail: abney.umich.edu.

Submission received: 26 August 2003; Revised submission received: 21 December 2003; Accepted for publication: 10 February 2004

Table 1
The Yarowsky algorithm variants. Y-1/DL-EM reduces H; the others reduce K.

Y-1/DL-EM-Λ	EM inner loop that uses labeled examples only
Y-1/DL-EM-X	EM inner loop that uses all examples
Y-1/DL-1-R	Near-original Yarowsky inner loop, no smoothing
Y-1/DL-1-VS	Near-original Yarowsky inner loop, "variable smoothing"
YS-P	Sequential update, "antismoothing"
YS-R	Sequential update, no smoothing
YS-FS	Sequential update, original Yarowsky smoothing

restricts the data sets on which the algorithm can be shown effective, but also for additional internal reasons. A detailed discussion would take us too far afield here, but suffice it to say that precision independence is a property that it would be preferable not to assume, but rather to derive from more basic properties of a data set, and that closer empirical study shows that precision independence fails to be satisfied in some data sets on which the Yarowsky algorithm is effective.

This article proposes a different approach. Instead of making assumptions about the data, it views the Yarowsky algorithm as optimizing an objective function. We will show that several variants of the algorithm (though not the algorithm in precisely its original form) optimize either negative log likelihood H or an alternative objective function, K, that imposes an upper bound on H.

Ideally, we would like to show that the Yarowsky algorithm minimizes H. Unfortunately, we are not able to do so. But we *are* able to show that a variant of the Yarowsky algorithm, which we call Y-1/DL-EM, decreases H in each iteration. It combines the outer loop of the Yarowsky algorithm with a different inner loop based on the expectation-maximization (EM) algorithm.

A second proposed variant of the Yarowsky algorithm, Y-1/DL-1, has the advantage that its inner loop is very similar to the original Yarowsky inner loop, unlike Y-1/DL-EM, whose inner loop bears little resemblance to the original. Y-1/DL-1 has the disadvantage that it does not directly reduce H, but we show that it does reduce the alternative objective function K.

We also consider a third variant, YS. It differs from Y-1/DL-EM and Y-1/DL-1 in that it updates sequentially (adding a single rule in each iteration), rather than in parallel (updating all rules in each iteration). Besides having the intrinsic interest of sequential update, YS can be proven effective when using exactly the same smoothing method as used in the original Yarowsky algorithm, in contrast to Y-1/DL-1, which uses either no smoothing or a nonstandard "variable smoothing." YS is proven to decrease K.

The Yarowsky algorithm variants that we consider are summarized in Table 1. To the extent that these variants capture the essence of the original algorithm, we have a better formal understanding of its effectiveness. Even if the variants are deemed to depart substantially from the original algorithm, we have at least obtained a family of new bootstrapping algorithms that are mathematically understood.

2. The Generic Yarowsky Algorithm

2.1 The Original Algorithm Y-0

The original Yarowsky algorithm, which we refer to as Y-0, is given in table 2. It is an iterative algorithm. One begins with a **seed set** $\Lambda^{(0)}$ of labeled examples and a

Table 2
The generic Yarowsky algorithm (Y-0)

(1) Given: examples X, and initial labeling $Y^{(0)}$
(2) For $t \in \{0, 1, \ldots\}$
  (2.1) Train classifier on labeled examples $(\Lambda^{(t)}, Y^{(t)})$, where $\Lambda^{(t)} = \{x \in X | Y^{(t)} \neq \perp\}$
    The resulting classifier predicts label j for example x with probability $\pi_x^{(t+1)}(j)$
  (2.2) For each example $x \in X$:
    (2.2.1) Set $\hat{y} = \arg\max_j \pi_x^{(t+1)}(j)$
    (2.2.2) Set $Y_x^{(t+1)} = \begin{cases} Y_x^{(0)} & \text{if } x \in \Lambda^{(0)} \\ \hat{y} & \text{if } \pi_x^{(t+1)}(\hat{y}) > \zeta \\ \perp & \text{otherwise} \end{cases}$
  (2.3) If $Y^{(t+1)} = Y^{(t)}$, stop

set $\mathrm{V}^{(0)}$ of unlabeled examples. At each iteration, a classifier is constructed from the labeled examples; then the classifier is applied to the unlabeled examples to create a new labeled set.

To discuss the algorithm formally, we require some notation. We assume first a set of examples X and a feature set F_x for each $x \in X$. The set of examples with feature f is X_f. Note that $x \in X_f$ if and only if $f \in F_x$.

We also require a series of labelings $Y^{(t)}$, where t represents the iteration number. We write $Y_x^{(t)}$ for the label of example x under labeling $Y^{(t)}$. An unlabeled example is one for which $Y_x^{(t)}$ is undefined, in which case we write $Y_x^{(t)} = \perp$. We write $\mathrm{V}^{(t)}$ for the set of unlabeled examples and $\Lambda^{(t)}$ for the set of labeled examples. It will also be useful to have a notation for the set of examples with label j:

$$\Lambda_j^{(t)} \equiv \{x \in X | Y_x^{(t)} = j \neq \perp\}$$

Note that $\Lambda^{(t)}$ is the disjoint union of the sets $\Lambda_j^{(t)}$. When t is clear from context, we drop the superscript (t) and write simply Λ_j, V, Y_x, etc.

At the risk of ambiguity, we will also sometimes write Λ_f for the set of labeled examples with feature f, trusting to the index to discriminate between Λ_f (labeled examples with feature f) and Λ_j (labeled examples with label j). We always use f and g to represent features and j and k to represent labels. The reader may wish to refer to Table 3, which summarizes notation used throughout the article.

In each iteration, the Yarowsky algorithm uses a supervised learner to train a classifier on the labeled examples. Let us call this supervised learner the **base learning algorithm**; it is a function from $(X, Y^{(t)})$ to a classifier π drawn from a space of classifiers Π. It is assumed that the classifier makes confidence-weighted predictions. That is, the classifier defines a scoring function $\pi(x, j)$, and the predicted label for example x is

$$\hat{y} \equiv \arg\max_j \pi(x, j) \tag{1}$$

Ties are broken arbitrarily. Technically, we assume a fixed order over labels and define the maximization as returning the first label in the ordering, in case of a tie.

It will be convenient to assume that the scoring function is nonnegative and bounded, in which case we can normalize it to make $\pi(x, j)$ a conditional distribution over labels j for a given example x. Henceforward, we write $\pi_x(j)$ instead of $\pi(x, j)$,

Table 3
Summary of notation.

X	set of examples, both labeled and unlabeled
Y	the current labeling; $Y^{(t)}$ is the labeling at iteration t
Λ	the (current) set of labeled examples
V	the (current) set of unlabeled examples
x	an example index
f, g	feature indices
j, k	label indices
F_x	the features of example x
Y_x	the label of example x; value is undefined ($\perp$) if x is unlabeled
X_f, Λ_f, V_f	examples, labeled examples, unlabeled examples that have feature f
Λ_j, Λ_{fj}	examples with label j, examples with feature f and label j
m	the number of features of a given example: $\lvert F_x \rvert$ (cf. equation (12))
L	the number of labels
$\phi_x(j)$	labeling distribution (equation (5))
$\pi_x(j)$	prediction distribution (equation (12); except for DL-0, which uses equation (11))
θ_{fj}	score for rule $f \rightarrow j$; we view θ_f as the prediction distribution of f
$\hat{y}$	label that maximizes $\pi_x(j)$ for given x (equation (1)
$[\![\Phi]\!]$	truth value of Φ: value is 0 or 1
H	objective function, negative log-likelihood (equation (6))
$H(p)$	entropy of distribution p
$H(p\|q)$	cross entropy: $-\sum_x p(x) \log q(x)$ (cf. equations (2) and (3))
K	objective function, upper bound on H (equation (20))
$q_f(j)$	precision of rule $f \rightarrow j$ (equation (9))
$\tilde{q}_f(j)$	smoothed precision (equation (10))
$\hat{q}_f(j)$	"peaked" precision (equation (25))
$j\dagger$	the label that maximizes precision $q_f(j)$ for a given feature f (equation (26))
$j*$	the label that maximizes rule score θ_{fj} for a given feature f (equation (28))
$u(\cdot)$	uniform distribution

understanding π_x to be a probability distribution over labels j. We call this distribution the **prediction distribution** of the classifier on example x.

To complete an iteration of the Yarowsky algorithm, one recomputes labels for examples. Specifically, the label $\hat{y}$ is assigned to example x if the score $\pi_x(\hat{y})$ exceeds a threshold ζ, called the **labeling threshold**. The new labeled set $\Lambda^{(t+1)}$ contains all examples for which $\pi_x(\hat{y}) > \zeta$. Relabeling applies only to examples in $\mathrm{V}^{(0)}$. The labels for examples in $\Lambda^{(0)}$ are indelible, because $\Lambda^{(0)}$ constitutes the original manually labeled data, as opposed to data that have been labeled by the learning algorithm itself.

The algorithm continues until convergence. The particular base learning algorithm that Yarowsky uses is deterministic, in the sense that the classifier induced is a deterministic function of the labeled data. Hence, the algorithm is known to have converged at whatever point the labeling remains unchanged.

Note that the algorithm as stated leaves the base learning algorithm unspecified. We can distinguish between the **generic Yarowsky algorithm** Y-0, for which the base learning algorithm is an open parameter, and the **specific Yarowsky algorithm**, which includes a specification of the base learner. Informally, we call the generic algorithm the outer loop and the base learner the inner loop of the specific Yarowsky algorithm. The base learner that Yarowsky assumes is a decision list induction algorithm. We postpone discussion of it until Section 3.

2.2 An Objective Function

Machine learning algorithms are typically designed to optimize some objective function that represents a formal measure of performance. The maximum-likelihood criterion is the most commonly used objective function. Suppose we have a set of examples Λ, with labels Y_x for $x \in \Lambda$, and a parametric family of models π_θ such that $\pi(j|x;\theta)$ represents the probability of assigning label j to example x, according to the model. The likelihood of θ is the probability of the full data set according to the model, viewed as a function of θ, and the maximum-likelihood criterion instructs us to choose the parameter settings $\hat{\theta}$ that maximize likelihood, or equivalently, log-likelihood:

$$\begin{aligned} l(\theta) &= \log \prod_{x\in\Lambda} \pi(Y_x|x;\theta) \\ &= \sum_{x\in\Lambda} \log \pi(Y_x|x;\theta) \\ &= \sum_{x\in\Lambda} \sum_j [\![j = Y_x]\!] \log \pi(j|x;\theta) \end{aligned}$$

(The notation $[\![\Phi]\!]$ represents the truth value of the proposition Φ; it is one if Φ is true and zero otherwise.)

Let us define

$$\phi_x(j) = [\![j = Y_x]\!] \qquad \text{for } x \in \Lambda$$

Note that ϕ_x satisfies the formal requirements of a probability distribution over labels j: Specifically, it is a point distribution with all its mass concentrated on Y_x. We call it the **labeling distribution.** Now we can write

$$\begin{aligned} l(\theta) &= \sum_{x\in\Lambda} \sum_j \phi_x(j) \log \pi(j|x;\theta) \\ &= -\sum_{x\in\Lambda} H(\phi_x \| \pi_x) \end{aligned} \tag{2}$$

In (2) we have written π_x for the distribution $\pi(\cdot|x;\theta)$, leaving the dependence on θ implicit. We have also used the nonstandard notation $H(p\|q)$ for what is sometimes called **cross entropy**. It is easy to verify that

$$H(p\|q) = H(p) + D(p\|q) \tag{3}$$

where $H(p)$ is the entropy of p and D is Kullback-Leibler divergence. Note that when p is a point distribution, $H(p) = 0$ and hence $H(p\|q) = D(p\|q)$. In particular:

$$l(\theta) = -\sum_{x\in\Lambda} D(\phi_x \| \pi_x) \tag{4}$$

Thus when, as here, ϕ_x is a point distribution, we can restate the maximum-likelihood criterion as instructing us to choose the model that minimizes the total divergence between the empirical labeling distributions ϕ_x and the model's prediction distributions π_x.

To extend $l(\theta)$ to unlabeled examples, we need only observe that unlabeled examples are ones about whose labels the data provide no information. Accordingly, we

revise the definition of ϕ_x to treat unlabeled examples as ones whose labeling distribution is the maximally uncertain distribution, which is to say, the uniform distribution:

$$\phi_x(j) = \begin{cases} [\![j = Y_x]\!] & \text{for } x \in \Lambda \\ \frac{1}{L} & \text{for } x \in \mathrm{V} \end{cases} \tag{5}$$

where L is the number of labels. Equivalently:

$$\phi_x(j) = [\![x \in \Lambda_j]\!] + [\![x \in \mathrm{V}]\!]\frac{1}{L}$$

When we replace Λ with X, expressions (2) and (4) are no longer equivalent; we must use (2). Since $H(\phi_x \| \pi_x) = H(\phi_x) + D(\phi_x \| \pi_x)$, and $H(\phi_x)$ is minimized when x is labeled, minimizing $H(\phi_x \| \pi_x)$ forces one to label unlabeled examples. On labeled examples, $H(\phi_x \| \pi_x) = D(\phi_x \| \pi_x)$, and $D(\phi_x \| \pi_x)$ is minimized when the labels of examples agree with the predictions of the model.

In short, we adopt as objective function

$$H \equiv \sum_{x \in X} H(\phi_x \| \pi_x) = -l(\phi, \theta) \tag{6}$$

We seek to minimize H.

2.3 The Modified Algorithm Y-1

We can show that a modified version of the Yarowsky algorithm finds a local minimum of H. Two modifications are necessary:

- The labeling function Y is recomputed in each iteration as before, but with the constraint that an example once labeled stays labeled. The label may change, but a labeled example cannot become unlabeled again.
- We eliminate the threshold ζ or (equivalently) fix it at $1/L$. As a result, the only examples that remain unlabeled after the labeling step are those for which π_x is the uniform distribution. The problem with an arbitrary threshold is that it prevents the algorithm from converging to a minimum of H. A threshold that gradually decreases to $1/L$ would also address the problem but would complicate the analysis.

The modified algorithm, Y-1, is given in Table 4.

To obtain a proof, it will be necessary to make an assumption about the supervised classifier $\pi^{(t+1)}$ induced by the base learner in step 2.1 of the algorithm. A natural assumption is that the base learner chooses $\pi^{(t+1)}$ so as to minimize $\sum_{x \in \Lambda^{(t)}} D(\phi_x^{(t)} \| \pi_x^{(t+1)})$. A weaker assumption will suffice, however. We assume that the base learner reduces divergence, if possible. That is, we assume

$$\Delta D_\Lambda \equiv \sum_{x \in \Lambda^{(t)}} D(\phi_x^{(t)} \| \pi_x^{(t+1)}) - \sum_{x \in \Lambda^{(t)}} D(\phi_x^{(t)} \| \pi_x^{(t)}) \leq 0 \tag{7}$$

with equality only if there is no classifier $\pi^{(t+1)} \in \Pi$ that makes $\Delta D_\Lambda < 0$. Note that any learning algorithm that minimizes $\sum_{x \in \Lambda^{(t)}} D(\phi_x^{(t)} \| \pi_x^{(t+1)})$ satisfies the weaker assumption (7), inasmuch as the option of setting $\pi_x^{(t+1)} = \pi_x^{(t)}$ is always available.

Table 4
The modified generic Yarowsky algorithm (Y-1).

(1) Given: $X, Y^{(0)}$
(2) For $t \in \{0, 1, \ldots\}$
  (2.1) Train classifier on $(\Lambda^{(t)}, Y^{(t)})$; result is $\pi^{(t+1)}$
  (2.2) For each example $x \in X$:
    (2.2.1) Set $\hat{y} = \arg\max_j \pi_x^{(t+1)}(j)$
    (2.2.2) Set $Y_x^{(t+1)} = \begin{cases} Y_x^{(0)} & \text{if } x \in \Lambda^{(0)} \\ \hat{y} & \text{if } x \in \Lambda^{(t)} \vee \pi_x^{(t+1)}(\hat{y}) > 1/L \\ \perp & \text{otherwise} \end{cases}$
  (2.3) If $Y^{(t+1)} = Y^{(t)}$, stop

We also consider a somewhat stronger assumption, namely, that the base learner reduces divergence over all examples, not just over labeled examples:

$$\Delta D_X \equiv \sum_{x \in X} D(\phi_x^{(t)} \| \pi_x^{(t+1)}) - \sum_{x \in X} D(\phi_x^{(t)} \| \pi_x^{(t)}) \leq 0 \tag{8}$$

If a base learning algorithm satisfies (8), the proof of theorem 1 is shorter; but (7) is the more natural condition for a base learner to satisfy.

We can now state the main theorem of this section.

Theorem 1
If the base learning algorithm satisfies (7) or (8), algorithm Y-1 decreases H at each iteration until it reaches a critical point of H.

We require the following lemma in order to prove the theorem:

Lemma 1
For all distributions p

$$H(p) \geq \log \frac{1}{\max_j p(j)}$$

with equality iff p is the uniform distribution.

Proof
By definition, for all k:

$$\begin{aligned} p(k) &\leq \max_j p(j) \\ \log \frac{1}{p(k)} &\geq \log \frac{1}{\max_j p(j)} \end{aligned}$$

Since this is true for all k, it is true if we take the expectation with respect to p:

$$\begin{aligned} \sum_k p(k) \log \frac{1}{p(k)} &\geq \sum_k p(k) \log \frac{1}{\max_j p(j)} \\ H(p) &\geq \log \frac{1}{\max_j p(j)} \end{aligned}$$

We have equality only if $p(k) = \max_j p(j)$ for all k, that is, only if p is the uniform distribution.

We now prove the theorem.

Proof of Theorem 1
The algorithm produces a sequence of labelings $\phi^{(0)}, \phi^{(1)}, \ldots$ and a sequence of classifiers $\pi^{(1)}, \pi^{(2)}, \ldots$. The classifier $\pi^{(t+1)}$ is trained on $\phi^{(t)}$, and the labeling $\phi^{(t+1)}$ is created using $\pi^{(t+1)}$.

Recall that

$$H = \sum_{x \in X} \left[H(\phi_x) + D(\phi_x \| \pi_x) \right]$$

In the training step (2.1) of the algorithm, we hold ϕ fixed and change π, and in the labeling step (2.2), we hold π fixed and change ϕ. We will show that the training step minimizes H as a function of π, and the labeling step minimizes H as a function of ϕ except in examples in which it is at a critical point of H. Hence, H is nonincreasing in each iteration of the algorithm and is strictly decreasing unless $(\phi^{(t)}, \pi^{(t)})$ is a critical point of H.

Let us consider the labeling step first. In this step, π is held constant, but ϕ (possibly) changes, and we have

$$\Delta H = \sum_{x \in X} \Delta H(x)$$

where

$$\Delta H(x) \equiv H(\phi_x^{(t+1)} \| \pi_x^{(t+1)}) - H(\phi_x^{(t)} \| \pi_x^{(t+1)})$$

We can show that ΔH is nonpositive if we can show that $\Delta H(x)$ is nonpositive for all x.

We can guarantee that $\Delta H(x) \leq 0$ if $\phi^{(t+1)}$ minimizes $H(p \| \pi_x^{(t+1)})$ viewed as a function of p. By definition:

$$H(p \| \pi_x^{(t+1)}) = \sum_j p_j \log \frac{1}{\pi_x^{(t+1)}(j)}$$

We wish to find the distribution p that minimizes $H(p \| \pi_x^{(t+1)})$. Clearly, we accomplish that by placing all the mass of p in p_{j*}, where $j*$ minimizes $-\log \pi_x^{(t+1)}(j)$. If there is more than one minimizer, $H(p \| \pi_x^{(t+1)})$ is minimized by any distribution p that distributes all its mass among the minimizers of $-\log \pi_x^{(t+1)}(j)$. Observe further that

$$\begin{aligned} \arg\min_j \log \frac{1}{\pi_x^{(t+1)}(j)} &= \arg\max_j \pi_x^{(t+1)}(j) \\ &= \hat{y} \end{aligned}$$

That is, we minimize $H(p \| \pi_x^{(t+1)})$ by setting $p_j = [\![j = \hat{y}]\!]$, which is to say, by labeling x as predicted by $\pi^{(t+1)}$. That is how algorithm Y-1 defines $\phi_x^{(t+1)}$ for all examples $x \in \Lambda^{(t+1)}$ whose labels are modifiable (that is, excluding $x \in \Lambda^{(0)}$).

Note that $\phi_x^{(t+1)}$ does not minimize $H(p \| \pi_x^{(t+1)})$ for examples $x \in \mathrm{V}^{(t+1)}$, that is, for examples x that remain unlabeled at $t+1$. However, in algorithm Y-1, any example that is unlabeled at $t+1$ is necessarily also unlabeled at t, so for any such example,

$\Delta H(x) = 0$. Hence, if any label changes in the labeling step, H decreases, and if no label changes, H remains unchanged; in either case, H does not increase.

We can show further that even for examples $x \in \mathrm{V}^{(t+1)}$, the labeling distribution $\phi_x^{(t+1)}$ assigned by Y-1 represents a critical point of H. For any example $x \in \mathrm{V}^{(t+1)}$, the prediction distribution $\pi_x^{(t+1)}$ is the uniform distribution (otherwise Y-1 would have labeled x). Hence the divergence between $\phi^{(t+1)}$ and $\pi^{(t+1)}$ is zero, and thus at a minimum. It would be possible to decrease $H(\phi_x^{(t+1)} \| \pi_x^{(t+1)})$ by decreasing $H(\phi_x^{(t+1)})$ at the cost of an increase in $D(\phi_x^{(t+1)} \| \pi_x^{(t+1)})$, but all directions of motion (all ways of selecting labels to receive increased probability mass) are equally good. That is to say, the gradient of H is zero; we are at a critical point.

Essentially, we have reached a saddle point. We have minimized H with respect to $\phi_x(j)$ along those dimensions with a nonzero gradient. Along the remaining dimensions, we are actually at a local maximum, but without a gradient to choose a direction of descent.

Now let us consider the algorithm's training step (2.1). In this step, ϕ is held constant, so the change in H is equal to the change in D—recall that $H(\phi \| \pi) = H(\phi) + D(\phi \| \pi)$. By the hypothesis of the theorem, there are two cases: The base learner satisfies either (7) or (8). If it satisfies (8), the base learner minimizes D as a function of π, hence it follows immediately that it minimizes H as a function of π.

Suppose instead that the base learner satisfies (7). We can express H as

$$H = \sum_{x \in X} H(\phi_x) + \sum_{x \in \Lambda^{(t)}} D(\phi_x \| \pi_x) + \sum_{x \in \mathrm{V}^{(t)}} D(\phi_x \| \pi_x)$$

In the training step, the first term remains constant. The second term decreases, by hypothesis. But the third term may increase. However, we can show that any increase in the third term is more than offset in the labeling step.

Consider an arbitrary example x in $\mathrm{V}^{(t)}$. Since it is unlabeled at time t, we know that $\phi_x^{(t)}$ is the uniform distribution u:

$$u(j) = \frac{1}{L}$$

Moreover, $\pi_x^{(t)}$ must also be the uniform distribution; otherwise example x would have been labeled in a previous iteration. Therefore the value of $H(x) = H(\phi_x \| \pi_x)$ at the beginning of iteration t is H_0:

$$H_0 = \sum_j \phi_x^{(t)}(j) \log \frac{1}{\pi_x^{(t)}(j)} = \sum_j u(j) \log \frac{1}{u(j)} = H(u)$$

After the training step, the value is H_1:

$$H_1 = \sum_j \phi_x^{(t)}(j) \log \frac{1}{\pi_x^{(t+1)}(j)}$$

If π_x remains unchanged in the training step, then the new distribution $\pi_x^{(t+1)}$, like the old one, is the uniform distribution, and the example remains unlabeled. Hence there is no change in H, and in particular, H is nonincreasing, as desired. On the other hand, if π_x does change, then the new distribution $\pi_x^{(t+1)}$ is nonuniform, and the example is

labeled in the labeling step. Hence the value of $H(x)$ at the end of the iteration, after the labeling step, is H_2:

$$H_2 = \sum_j \phi_x^{(t+1)}(j) \log \frac{1}{\pi_x^{(t+1)}(j)} = \log \frac{1}{\pi_x^{(t+1)}(\hat{y})}$$

By Lemma 1, $H_2 < H(u)$; hence $H_2 < H_0$.

As we observed above, $H_1 > H_0$, but if we consider the change overall, we find that the increase in the training step is more than offset in the labeling step:

$$\Delta H(x) = H_2 - H_1 + H_1 - H_0 < 0$$

3. The Specific Yarowsky Algorithm

3.1 The Original Decision List Induction Algorithm DL-0

When one speaks of the Yarowsky algorithm, one often has in mind not just the generic algorithm Y-0 (or Y-1), but an algorithm whose specification includes the particular choice of base learning algorithm made by Yarowsky. Specifically, Yarowsky's base learner constructs a decision list, that is, a list of rules of form $f \rightarrow j$, where f is a feature and j is a label, with score θ_{fj}. A rule $f \rightarrow j$ matches example x if x possesses the feature f. The label predicted for a given example x is the label of the highest scoring rule that matches x.

Yarowsky uses smoothed precision for rule scoring. As the name suggests, smoothed precision $\tilde{q}_f(j)$ is a smoothed version of (raw) precision $q_f(j)$, which is the probability that rule $f \rightarrow j$ is correct given that it matches

$$q_f(j) \equiv \begin{cases} |\Lambda_{fj}|/|\Lambda_f| & \text{if } |\Lambda_f| > 0 \\ 1/L & \text{otherwise} \end{cases} \tag{9}$$

where Λ_f is the set of labeled examples that possess feature f, and Λ_{fj} is the set of labeled examples with feature f and label j.

Smoothed precision $\tilde{q}(j|f;\epsilon)$ is defined as follows:

$$\tilde{q}(j|f;\epsilon) \equiv \frac{|\Lambda_{fj}| + \epsilon}{|\Lambda_f| + L\epsilon} \tag{10}$$

We also write $\tilde{q}_f(j)$ when ϵ is clear from context.

Yarowsky defines a rule's score to be its smoothed precision:

$$\theta_{fj} = \tilde{q}_f(j)$$

Anticipating later needs, we will also consider raw precision as an alternative: $\theta_{fj} = q_f(j)$. Both raw and smoothed precision have the properties of a conditional probability distribution. Generally, we view θ_{fj} as a conditional distribution over labels j for a fixed feature f.

Yarowsky defines the confidence of the decision list to be the score of the highest-scoring rule that matches the instance being classified. This is equivalent to defining

$$\pi_x(j) \propto \max_{f \in F_x} \theta_{fj} \tag{11}$$

(Recall that F_x is the set of features of x.) Since the classifier's prediction for x is defined, in equation (1), to be the label that maximizes $\pi_x(j)$, definition (11) implies

Table 5
The decision list induction algorithm DL-0. The value accumulated in $N[f,j]$ is $|\Lambda_{fj}|$, and the value accumulated in $Z[f]$ is $|\Lambda_f|$.

(0) Given: a fixed value for $\epsilon > 0$
  Initialize arrays $N[f,j] = 0$, $Z[f] = 0$ for all f, j
(1) For each example $x \in \Lambda$
  (1.1) Let j be the label of x
  (1.2) Increment $N[f,j]$, $Z[f]$, for each feature f of x
(2) For each feature f and label j
  (2.1) Set $\theta_{fj} = \frac{N[f,j]+\epsilon}{Z[f]+L\epsilon}$
(*) Define $\pi_x(j) \propto \max_{f \in F_x} \theta_{fj}$

that the classifier's prediction is the label of the highest-scoring rule matching x, as desired.

We have written $\propto$ in (11) rather than = because maximizing θ_{fj} across $f \in F_x$ for each label j will not in general yield a probability distribution over labels—though the scores will be positive and bounded, and hence normalizable. Considering only the final predicted label $\hat{y}$ for a given example x, the normalization will have no effect, inasmuch as all scores θ_{fj} being compared will be scaled in the same way.

As characterized by Yarowsky, a decision list contains only those rules $f \to j$ whose score $\tilde{q}_f(j)$ exceeds the labeling threshold ζ. This can be seen purely as an efficiency measure. Including rules whose score falls below the labeling threshold will have no effect on the classifier's predictions, as the threshold will be applied when the classifier is applied to examples. For this reason, we do not prune the list. That is, we represent a decision list as a set of parameters $\{\theta_{fj}\}$, one for every possible rule $f \to j$ in the cross product of the set of features and the set of labels.

The decision list induction algorithm used by Yarowsky is summarized in Table 5; we refer to it as DL-0. Note that the step labeled (*) is not actually a step of the induction algorithm but rather specifies how the decision list is used to compute a prediction distribution π_x for a given example x.

Unfortunately, we cannot prove anything about DL-0 as it stands. In particular, we are unable to show that DL-0 reduces divergence between prediction and labeling distributions (7). In the next section, we describe an alternative decision list induction algorithm, DL-EM, that does satisfy (7); hence we can apply Theorem 1 to the combination Y-1/DL-EM to show that it reduces H. However, a disadvantage of DL-EM is that it does not resemble the algorithm DL-0 used by Yarowsky. We return in section 3.4 to a close variant of DL-0 called DL-1 and show that though it does not directly reduce H, it does reduce the upper bound K.

3.2 The Decision List Induction Algorithm DL-EM

The algorithm DL-EM is a special case of the EM algorithm. We consider two versions of the algorithm: DL-EM-Λ and DL-EM-X. They differ in that DL-EM-Λ is trained on labeled examples only, whereas DL-EM-X is trained on both labeled and unlabeled examples. However, the basic outline of the algorithm is the same for both.

First, the DL-EM algorithms do not assume Yarowsky's definition of π, given in (11). As discussed above, the parameters θ_{fj} can be thought of as defining a prediction distribution $\theta_f(j)$ over labels j for each feature f. Hence equation (11) specifies how the prediction distributions θ_f for the features of example x are to be combined to yield a

prediction distribution π_x for x. Instead of combining distributions by maximizing θ_{fj} across $f \in F_x$ as in equation (11), DL-EM takes a mixture of the θ_f:

$$\pi_x(j) = \frac{1}{m} \sum_{f \in F_x} \theta_{fj} \tag{12}$$

Here $m = |F_x|$ is the number of features that x possesses; for the sake of simplicity, we assume that all examples have the same number of features. Since θ_f is a probability distribution for each f, and since any convex combination of distributions is also a distribution, it follows that π_x as defined in (12) is a probability distribution.

The two definitions for $\pi_x(j)$, (11) and (12), will often have the same mode $\hat{y}$, but that is guaranteed only in the rather severely restricted case of two features and two labels. Under definition (11), the prediction is determined entirely by the strongest θ_f, whereas definition (12) permits a bloc of weaker θ_f to outvote the strongest one. Yarowsky explicitly wished to avoid the possibility of such interactions. Nonetheless, definition (12), used by DL-EM, turns out to make analysis of other base learners more manageable, and we will assume it henceforth, not only for DL-EM, but also for the algorithms DL-1 and YS discussed in subsequent sections.

DL-EM also differs from DL-0 in that DL-EM does not construct a classifier "from scratch" but rather seeks to improve on a previous classifier. In the context of the Yarowsky algorithm, the previous classifier is the one from the previous iteration of the outer loop. We write θ_{fj}^{old} for the parameters and π_x^{old} for the prediction distributions of the previous classifier.

Conceptually, DL-EM considers the label j assigned to an example x to be generated by choosing a feature $f \in F_x$ and then assigning the label j according to the feature's prediction distribution $\theta_f(j)$. The choice of feature f is a hidden variable. The degree to which an example labeled j is imputed to feature f is determined by the old distribution:

$$\pi^{\text{old}}(f|x,j) = \frac{[\![f \in F_x]\!]\theta_{fj}^{\text{old}}}{\sum_g [\![g \in F_x]\!]\theta_{gj}^{\text{old}}} = \frac{[\![f \in F_x]\!]\frac{1}{m}\theta_{fj}^{\text{old}}}{\pi_x^{\text{old}}(j)}$$

One can think of $\pi^{\text{old}}(f|x,j)$ either as the posterior probability that feature f was responsible for the label j, or as the portion of the labeled example (x,j) that is imputed to feature f. We also write $\pi_{xj}^{\text{old}}(f)$ as a synonym for $\pi^{\text{old}}(f|x,j)$. The new estimate θ_{fj} is obtained by summing imputed occurrences of (f,j) and normalizing across labels. For DL-EM-Λ, this takes the form

$$\theta_{fj} = \frac{\sum_{x \in \Lambda_j} \pi^{\text{old}}(f|x,j)}{\sum_k \sum_{x \in \Lambda_k} \pi^{\text{old}}(f|x,k)}$$

The algorithm is summarized in Table 6.

The second version of the algorithm, DL-EM-X, is summarized in Table 7. It is like DL-EM-Λ, except that it uses the update rule

$$\theta_{fj} = \frac{\sum_{x \in \Lambda_j} \pi^{\text{old}}(f|x,j) + \frac{1}{L}\sum_{x \in V} \pi^{\text{old}}(f|x,j)}{\sum_k \left[\sum_{x \in \Lambda_k} \pi^{\text{old}}(f|x,k) + \frac{1}{L}\sum_{x \in V} \pi^{\text{old}}(f|x,k)\right]} \tag{13}$$

Update rule (13) includes unlabeled examples as well as labeled examples. Conceptually, it divides each unlabeled example equally among the labels, then divides the resulting fractional labeled example among the example's features.

Table 6
DL-EM-Λ decision list induction algorithm.

(0) Initialize $N[f,j] = 0$ for all f, j
(1) For each example x labeled j
  (1.1) Let $Z = \sum_{g \in F_x} \theta^{\text{old}}_{gj}$
  (1.2) For each $f \in F_x$, increment $N[f,j]$ by $\frac{1}{Z}\theta^{\text{old}}_{fj}$
(2) For each feature f
  (2.1) Let $Z = \sum_j N[f,j]$
  (2.2) For each label j, set $\theta_{fj} = \frac{1}{Z}N[f,j]$

Table 7
DL-EM-X decision list induction algorithm.

(0) Initialize $N[f,j] = 0$ and $U[f,j] = 0$, for all f, j
(1) For each example x labeled j
  (1.1) Let $Z = \sum_{g \in F_x} \theta^{\text{old}}_{gj}$
  (1.2) For each $f \in F_x$, increment $N[f,j]$ by $\frac{1}{Z}\theta^{\text{old}}_{fj}$
(2) For each unlabeled example x
  (2.1) Let $Z = \sum_{g \in F_x} \theta^{\text{old}}_{gj}$
  (2.2) For each $f \in F_x$, increment $U[f,j]$ by $\frac{1}{Z}\theta^{\text{old}}_{fj}$
(3) For each feature f
  (3.1) Let $Z = \sum_j (N[f,j] + \frac{1}{L}U[f,j])$
  (3.2) For each label j, set $\theta_{fj} = \frac{1}{Z}\left(N[f,j] + \frac{1}{L}U[f,j]\right)$

We note that both variants of the DL-EM algorithm constitute a single iteration of an EM-like algorithm. A single iteration suffices to prove the following theorem, though multiple iterations would also be effective:

Theorem 2
The classifier produced by the DL-EM-Λ algorithm satisfies equation (7), and the classifier produced by the DL-EM-X algorithm satisfies equation (8).

Combining Theorems 1 and 2 yields the following corollary:

Corollary
The Yarowsky algorithm Y-1, using DL-EM-Λ or DL-EM-X as its base learning algorithm, decreases H at each iteration until it reaches a critical point of H.

Proof of Theorem 2
Let θ^{old} represent the parameter values at the beginning of the call to DL-EM, let θ represent a family of free variables that we will optimize, and let π^{old} and π be the corresponding prediction distributions. The labeling distribution ϕ is fixed. For any set of examples α, let ΔD_α be the change in $\sum_{x \in \alpha} D(\phi_x \| \pi_x)$ resulting from the change in θ. We are obviously particularly interested in two cases: that in which α is the set of all examples X (for DL-EM-X) and that in which α is the set of labeled examples

Λ (for DL-EM-Λ). In either case, we will show that $\Delta D_\alpha \leq 0$, with equality only if no choice of θ decreases D.

We first derive an expression for $-\Delta D_\alpha$ that we will put to use shortly:

$$\begin{aligned} -\Delta D_\alpha &= \sum_{x\in\alpha} \Big[D(\phi_x \| \pi_x^{\text{old}}) - D(\phi_x \| \pi_x) \Big] \\ &= \sum_{x\in\alpha} \Big[H(\phi_x \| \pi_x^{\text{old}}) - H(\phi_x) - H(\phi_x \| \pi_x) + H(\phi_x) \Big] \\ &= \sum_{x\in\alpha} \sum_j \phi_x(j) \Big[\log \pi_x(j) - \log \pi_x^{\text{old}}(j) \Big] \end{aligned} \tag{14}$$

The EM algorithm is based on the fact that divergence is non-negative, and strictly positive if the distributions compared are not identical:

$$\begin{aligned} 0 &\leq \sum_j \sum_{x\in\alpha} \phi_x(j) D(\pi_{xj}^{\text{old}} \| \pi_{xj}) \\ &= \sum_j \sum_{x\in\alpha} \phi_x(j) \sum_{f\in F_x} \pi_{xj}^{\text{old}}(f) \log \frac{\pi_{xj}^{\text{old}}(f)}{\pi_{xj}(f)} \\ &= \sum_j \sum_{x\in\alpha} \phi_x(j) \sum_{f\in F_x} \pi_{xj}^{\text{old}}(f) \log \left(\frac{\theta_{fj}^{\text{old}}}{\pi_x^{\text{old}}(j)} \cdot \frac{\pi_x(j)}{\theta_{fj}} \right) \end{aligned}$$

which yields the inequality

$$\sum_j \sum_{x\in\alpha} \phi_x(j) \Big[\log \pi_x(j) - \log \pi_x^{\text{old}}(j) \Big] \geq \sum_j \sum_{x\in\alpha} \phi_x(j) \sum_{f\in F_x} \pi_{xj}^{\text{old}}(f) \Big[\log \theta_{fj} - \log \theta_{fj}^{\text{old}} \Big]$$

By (14), this can be written as

$$-\Delta D_\alpha \geq \sum_j \sum_{x\in\alpha} \phi_x(j) \sum_{f\in F_x} \pi_{xj}^{\text{old}}(f) \Big[\log \theta_{fj} - \log \theta_{fj}^{\text{old}} \Big] \tag{15}$$

Since θ_{fj}^{old} is constant, by maximizing

$$\sum_j \sum_{x\in\alpha} \phi_x(j) \sum_{f\in F_x} \pi_{xj}^{\text{old}}(f) \log \theta_{fj} \tag{16}$$

we maximize a lower bound on $-\Delta D_\alpha$. It is easy to see that $-\Delta D_\alpha$ is bounded above by zero: we simply set $\theta_{fj} = \theta_{fj}^{\text{old}}$. Since divergence is zero only if the two distributions are identical, we have strict inequality in (15) unless the best choice for θ is θ^{old}, in which case no choice of θ makes $\Delta D_\alpha < 0$.

It remains to show that DL-EM computes the parameter set θ that maximizes (16). We wish to maximize (16) under the constraints that the values $\{\theta_{fj}\}$ for fixed f sum to unity across choices of j, so we apply Lagrange's method. We express the constraints in the form

$$C_f = 0$$

where

$$C_f \equiv \sum_j \theta_{fj} - 1$$

We seek a solution to the family of equations that results from expressing the gradient of (16) as a linear combination of the gradients of the constraints:

$$\frac{\partial}{\partial \theta_{fj}} \sum_k \sum_{x \in \alpha} \phi_x(k) \sum_{g \in F_x} \pi_{xk}^{\text{old}}(g) \log \theta_{gk} = \lambda_f \frac{\partial C_f}{\partial \theta_{fj}} \tag{17}$$

We derive an expression for the derivative on the left-hand side:

$$\frac{\partial}{\partial \theta_{fj}} \sum_k \sum_{x \in \alpha} \phi_x(k) \sum_{g \in F_x} \pi_{xk}^{\text{old}}(g) \log \theta_{gk} = \sum_{x \in X_f \cap \alpha} \phi_x(j) \pi_{xj}^{\text{old}}(f) \frac{1}{\theta_{fj}}$$

Similarly for the right-hand side:

$$\frac{\partial C_f}{\partial \theta_{fj}} = 1$$

Substituting these into equation (17):

$$\sum_{x \in X_f \cap \alpha} \phi_x(j) \pi_{xj}^{\text{old}}(f) \frac{1}{\theta_{fj}} = \lambda_f$$

$$\theta_{fj} = \sum_{x \in X_f \cap \alpha} \phi_x(j) \pi_{xj}^{\text{old}}(f) \frac{1}{\lambda_f} \tag{18}$$

Using the constraint $C_f = 0$ and solving for λ_f:

$$\sum_j \sum_{x \in X_f \cap \alpha} \phi_x(j) \pi_{xj}^{\text{old}}(f) \frac{1}{\lambda_f} - 1 = 0$$

$$\lambda_f = \sum_j \sum_{x \in X_f \cap \alpha} \phi_x(j) \pi_{xj}^{\text{old}}(f)$$

Substituting back into (18):

$$\theta_{fj} = \frac{\sum_{x \in X_f \cap \alpha} \phi_x(j) \pi_{xj}^{\text{old}}(f)}{\sum_k \sum_{x \in X_f \cap \alpha} \phi_x(k) \pi_{xk}^{\text{old}}(f)} \tag{19}$$

If we consider the case where α is the set of all examples and expand ϕ_x in (19), we obtain

$$\theta_{fj} = \frac{1}{Z} \left[\sum_{x \in \Lambda_{fj}} \pi_{xj}^{\text{old}}(f) + \frac{1}{L} \sum_{x \in V_f} \pi_{xj}^{\text{old}}(f) \right]$$

where Z normalizes θ_f. It is not hard to see that this is the update rule that DL-EM-X computes, using the intermediate values:

$$\begin{aligned} N[f,j] &= \sum_{x \in \Lambda_{fj}} \pi_{xj}^{\text{old}}(f) \\ U[f,j] &= \sum_{x \in V_f} \pi_{xj}^{\text{old}}(f) \end{aligned}$$

If we consider the case where α is the set of labeled examples and expand ϕ_x in (19), we obtain

$$\theta_{fj} = \frac{1}{Z} \sum_{x \in \Lambda_{fj}} \pi_{xj}^{\text{old}}(f)$$

This is the update rule that DL-EM-Λ computes. Thus we see that DL-EM-X reduces D_X, and DL-EM-Λ reduces D_Λ.

We note in closing that DL-EM-X can be simplified when used with algorithm Y-1, inasmuch as it is known that $\theta_{fj} = 1/L$ for all (f, j), where $f \in F_x$ for some $x \in \text{V}$. Then the expression for $U[f, j]$ simplifies as follows:

$$\begin{aligned} & \sum_{x \in \text{V}_f} \pi_{xj}^{\text{old}}(f) \\ &= \sum_{x \in \text{V}_f} \left[\frac{1/L}{\sum_{g \in F_x} 1/L} \right] \\ &= \frac{|\text{V}_f|}{m} \end{aligned}$$

The dependence on j disappears, so we can replace $U[f, j]$ with $U[f]$ in algorithm DL-EM-X, delete step 2.1, and replace step 2.2 with the statement "For each $f \in F_x$, increment $U[f]$ by $1/m$."

3.3 The Objective Function K

Y-1/DL-EM is the only variation on the Yarowsky algorithm that we can show to reduce negative log-likelihood, H. The variants that we discuss in the remainder of the article, Y-1/DL-1 and YS, reduce an alternative objective function, K, which we now define.

The value K (or, more precisely, the value K/m) is an upper bound on H, which we derive using Jensen's inequality, as follows:

$$\begin{aligned} H &= -\sum_{x \in X} \sum_j \phi_{xj} \log \sum_{g \in F_x} \frac{1}{m} \theta_{gj} \\ &\leq -\sum_{x \in X} \sum_j \phi_{xj} \sum_{g \in F_x} \frac{1}{m} \log \theta_{gj} \\ &= \frac{1}{m} \sum_{x \in X} \sum_{g \in F_x} H(\phi_x \| \theta_g) \end{aligned}$$

We define

$$K \equiv \sum_{x \in X} \sum_{g \in F_x} H(\phi_x \| \theta_g) \tag{20}$$

By minimizing K, we minimize an upper bound on H. Moreover, it is in principle possible to reduce K to zero. Since $H(\phi_x \| \theta_g) = H(\phi_x) + D(\phi_x \| \theta_g)$, K is reduced to zero if all examples are labeled, each feature concentrates its prediction distribution in a single label, and the label of every example agrees with the prediction of every feature it possesses. In this limiting case, any minimizer of K is also a minimizer of H.

Table 8
The decision list induction algorithm DL-1-R.

(0) Initialize $N[f,j] = 0$, $Z[f] = 0$ for all f, j
(1) For each example-label pair (x,j)
  (1.1) For each feature $f \in F_x$, increment $N[f,j]$, $Z[f]$
(2) For each feature f and label j
  (2.1) Set $\theta_{fj} = \frac{N[f,j]}{Z[f]}$
(*) Define $\pi_x(j) = \frac{1}{m} \sum_{f \in F_x} \theta_{fj}$

Table 9
The decision list induction algorithm DL-1-VS.

(0) Initialize $N[f,j] = 0$, $Z[f] = 0$, $U[f] = 0$ for all f, j
(1) For each example-label pair (x,j)
  (1.1) For each feature $f \in F_x$, increment $N[f,j]$, $Z[f]$
(2) For each unlabeled example x
  (2.1) For each feature $f \in F_x$, increment $U[f]$
(3) For each feature f and label j
  (3.1) Set $\epsilon = U[f]/L$
  (3.2) Set $\theta_{fj} = \frac{N[f,j]+\epsilon}{Z[f]+U[f]}$
(*) Define $\pi_x(j) = \frac{1}{m} \sum_{f \in F_x} \theta_{fj}$

We hasten to add a proviso: It is not possible to reduce K to zero for all data sets. The following provides a necessary and sufficient condition for being able to do so. Consider an undirected bipartite graph G whose nodes are examples and features. There is an edge between example x and feature f just in case f is a feature of x. Define examples x_1 and x_2 to be neighbors if they both belong to the same connected component of G. K is reducible to zero if and only if x_1 and x_2 have the same label according to $Y^{(0)}$, for all pairs of neighbors x_1 and x_2 in $\Lambda^{(0)}$.

3.4 Algorithm DL-1

We consider two variants of DL-0, called DL-1-R and DL-1-VS. They differ from DL-0 in two ways. First, the DL-1 algorithms assume the "mean" definition of π_x given in equation (12) rather than the "max" definition of equation (11). This is not actually a difference in the induction algorithm itself, but in the way the decision list is used to construct a prediction distribution π_x.

Second, the DL-1 algorithms use update rules that differ from the smoothed precision of DL-0. DL-1-R (Table 8) uses raw precision instead of smoothed precision. DL-1-VS (Table 9) uses smoothed precision, but unlike DL-0, DL-1-VS does not use a fixed smoothing constant ϵ; rather ϵ varies from feature to feature. Specifically, in computing the score θ_{fj}, DL-1-VS uses $|V_f|/L$ as its value for ϵ.

The value of ϵ used by DL-1-VS can be expressed in another way that will prove useful. Let us define

$$p(\Lambda|f) \equiv \frac{|\Lambda_f|}{|X_f|}$$

$$p(V|f) \equiv \frac{|V_f|}{|X_f|}$$

Lemma 2
The parameter values $\{\theta_{fj}\}$ computed by DL-1-VS can be expressed as

$$\theta_{fj} = p(\Lambda|f)q_f(j) + p(\mathrm{V}|f)u(j) \tag{21}$$

where $u(j)$ is the uniform distribution over labels.

Proof
If $|\Lambda_f| = 0$, then $p(\Lambda|f) = 0$ and $\theta_{fj} = u(j)$. Further, $N[f,j] = Z[f] = 0$, so DL-1-VS computes $\theta_{fj} = u(j)$, and the lemma is proved. Hence we need only consider the case $|\Lambda_f| > 0$.

First we show that smoothed precision can be expressed as a convex combination of raw precision (9) and the uniform distribution. Define $\delta = \epsilon/|\Lambda_f|$. Then:

$$\begin{aligned}
\tilde{q}_f(j) &= \frac{|\Lambda_{fj}| + \epsilon}{|\Lambda_f| + L\epsilon} \\
&= \frac{|\Lambda_{fj}|/|\Lambda_f| + \delta}{1 + L\delta} \\
&= \frac{1}{1 + L\delta} q_f(j) + \frac{L\delta}{1 + L\delta} \cdot \frac{\delta}{L\delta} \\
&= \frac{1}{1 + L\delta} q_f(j) + \frac{L\delta}{1 + L\delta} u(j)
\end{aligned} \tag{22}$$

Now we show that the mixing coefficient $1/(1 + L\delta)$ of (22) is the same as the mixing coefficient $p(\Lambda|f)$ of the lemma, when $\epsilon = |\mathrm{V}_f|/L$ as in step 3.1 of DL-1-VS:

$$\begin{aligned}
\epsilon &= \frac{|\mathrm{V}_f|}{L} \\
&= \frac{|\Lambda_f|}{L} \cdot \frac{p(\mathrm{V}|f)}{p(\Lambda|f)} \\
L\delta &= \frac{1}{p(\Lambda|f)} - 1 \\
\frac{1}{1 + L\delta} &= p(\Lambda|f)
\end{aligned}$$

The main theorem of this section (Theorem 5) is that the specific Yarowsky algorithm Y-1/DL-1 decreases K in each iteration until it reaches a critical point. It is proved as a corollary of two theorems. The first (Theorem 3) shows that DL-1 minimizes K as a function of θ, holding ϕ constant, and the second (Theorem 4) shows that Y-1 decreases K as a function of ϕ, holding θ constant. More precisely, DL-1-R minimizes K over labeled examples Λ, and DL-1-VS minimizes K over all examples X. Either is sufficient for Y-1 to be effective.

Theorem 3
DL-1 minimizes K as a function of θ, holding ϕ constant. Specifically, DL-1-R minimizes K over labeled examples Λ, and DL-1-VS minimizes K over all examples X.

Proof
We wish to minimize K as a function of θ under the constraints

$$C_f \equiv \sum_j \theta_{fj} - 1 = 0$$

for each f. As before, to minimize K under the constraints $C_f = 0$, we express the gradient of K as a linear combination of the gradients of the constraints and solve the resulting system of equations:

$$\frac{\partial K}{\partial \theta_{fj}} = \lambda_f \frac{\partial C_f}{\partial \theta_{fj}} \tag{23}$$

First we derive expressions for the derivatives of C_f and K. The variable α represents the set of examples over which we are minimizing K:

$$\frac{\partial C_f}{\partial \theta_{fj}} = 1$$

$$\begin{aligned} \frac{\partial K}{\partial \theta_{fj}} &= -\frac{\partial}{\partial \theta_{fj}} \sum_{x \in \alpha} \sum_{g \in F_x} \sum_k \phi_{xk} \log \theta_{gk} \\ &= -\sum_{x \in X_f \cap \alpha} \phi_{xj} \frac{1}{\theta_{fj}} \end{aligned}$$

We substitute these expressions into (23) and solve for θ_{fj}:

$$-\sum_{x \in X_f \cap \alpha} \phi_{xj} \frac{1}{\theta_{fj}} = \lambda_f$$

$$\theta_{fj} = -\sum_{x \in X_f \cap \alpha} \phi_{xj} / \lambda_f$$

Substituting the latter expression into the equation for $C_f = 0$ and solving for f yields

$$\sum_j \left(-\sum_{x \in X_f \cap \alpha} \phi_{xj} / \lambda_f \right) = 1$$

$$-|X_f \cap \alpha| = \lambda_f$$

Substituting this back into the expression for θ_{fj} gives us

$$\theta_{fj} = \frac{1}{|X_f \cap \alpha|} \sum_{x \in X_f \cap \alpha} \phi_{xj} \tag{24}$$

If $\alpha = \Lambda$, we have

$$\begin{aligned} \theta_{fj} &= \frac{1}{|\Lambda_f|} \sum_{x \in \Lambda_f} [\![x \in \Lambda_j]\!] \\ &= q_f(j) \end{aligned}$$

This is the update computed by DL-1-R, showing that DL-1-R computes the parameter values $\{\theta_{fj}\}$ that minimize K over the labeled examples Λ.

If $\alpha = X$, we have

$$\begin{aligned}\theta_{fj} &= \frac{1}{|X_f|}\sum_{x\in\Lambda_f}[\![x\in\Lambda_j]\!] + \frac{1}{|X_f|}\sum_{x\in V_f}\frac{1}{L} \\ &= \frac{|\Lambda_f|}{|X_f|}\cdot\frac{|\Lambda_{fj}|}{|\Lambda_f|} + \frac{|V_f|}{|X_f|}\cdot\frac{1}{L} \\ &= p(\Lambda|f)\cdot q_f(j) + p(V|f)\cdot u(j)\end{aligned}$$

By Lemma 2, this is the update computed by DL-1-VS, hence DL-1-VS minimizes K over the complete set of examples X.

Theorem 4
If the base learner decreases K over X or over Λ, where the prediction distribution is computed as

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

then algorithm Y-1 decreases K at each iteration until it reaches a critical point, considering K as a function of ϕ with θ held constant.

Proof
The proof has the same structure as the proof of Theorem 1, so we give only a sketch here. We minimize K as a function of ϕ by minimizing it for each example separately:

$$\begin{aligned}K(x) &= \sum_{g\in F_x} H(\phi_x\|\theta_g) \\ &= \sum_j \phi_{xj}\sum_{g\in F_x}\log\frac{1}{\theta_{gj}}\end{aligned}$$

To minimize $K(x)$, we choose ϕ_{xj} so as to concentrate all mass in

$$\arg\min_j \sum_{g\in F_x}\log\frac{1}{\theta_{gj}} = \arg\max_j \pi_x(j)$$

This is the labeling rule used by Y-1.

If the base learner minimizes over Λ only, rather than X, it can be shown that any increase in K on unlabeled examples is compensated for in the labeling step, as in the proof of Theorem 1.

Theorem 5
The specific Yarowsky algorithms Y-1/DL-1-R and Y-1/DL-1-VS decrease K at each iteration until they reach a critical point.

Proof
Immediate from Theorems 3 and 4.

4. Sequential Algorithms

4.1 The Family YS

The Yarowsky algorithm variants we have considered up to now do "parallel" updates in the sense that the parameters $\{\theta_{fj}\}$ are completely recomputed at each iteration. In

this section, we consider a family YS of "sequential" variants of the Yarowsky algorithm, in which a single feature is selected for update at each iteration. The YS algorithms resemble the "Yarowsky-Cautious" algorithm of Collins & Singer (1999), though they differ from that algorithm in that they update a single feature in each iteration, rather than a small set of features, as in Yarowsky-Cautious. The YS algorithms are intended to be as close to the Y-1/DL-1 algorithm as is consonant with single-feature updates. The YS algorithms differ from one another, and from Y-1/DL-1, in the choice of update rule. An interesting range of update rules work in the sequential setting. In particular, smoothed precision with fixed ϵ, as in the original algorithm Y-0/DL-0, works in the sequential setting, though with a proviso that will be spelled out later.

Instead of an initial labeled set, there is an initial classifier consisting of a set of selected features S_0 and initial parameter set $\theta^{(0)}$ with $\theta^{(0)}_{fj} = 1/L$ for all $f \notin S_0$. At each iteration, one feature is selected to be added to the selected set. A feature, once selected, remains in the selected set. It is permissible for a feature to be selected more than once; this permits us to continue reducing K even after all features have been selected. In short, there is a sequence of selected features $f_0, f_1, \ldots$, and

$$S_{t+1} = S_t \cup \{f_t\}$$

The parameters for the selected feature are also updated. At iteration t, the parameters θ_{gj}, with $g = f_t$, may be modified, but all other parameters remain constant. That is:

$$\theta^{(t+1)}_{gj} = \theta^{(t)}_{gj} \quad \text{for } g \neq f_t$$

It follows that, for all t:

$$\theta^{(t)}_{gj} = \frac{1}{L} \quad \text{for } g \notin S_t$$

However, parameters for features in S_0 may not be modified, inasmuch as they play the role of manually labeled data.

In each iteration, one selects a feature f_t and computes (or recomputes) the prediction distribution θ_{f_t} for the selected feature f_t. Then labels are recomputed as follows. Recall that $\hat{y} \equiv \arg\max_j \pi_x(j)$, where we continue to assume $\pi_x(j)$ to have the "mixture" definition (equation (12)). The label of example x is set to $\hat{y}$ if any feature of x belongs to S_{t+1}. In particular, all previously labeled examples continue to be labeled (though their labels may change), and any unlabeled examples possessing feature f_t become labeled.

The algorithm is summarized in Table 10. It is actually an algorithm schema; the definition for "update" needs to be supplied. We consider three different update functions: one that uses raw precision as its prediction distribution, one that uses smoothed precision, and one that goes in the opposite direction, using what we might call "peaked precision." As we have seen, smoothed precision can be expressed as a mixture of raw precision and the uniform (i.e., maximum-entropy) distribution (22). Peaked precision $\hat{q}(f)$ mixes in a certain amount of the point (i.e., minimum-entropy) distribution that has all its mass on the label that maximizes raw precision:

$$\hat{q}_f(j) \equiv p(\Lambda^{(t)}|f) q_f(j) + p(\mathrm{V}^{(t)}|f) [\![j = j\dagger]\!] \tag{25}$$

where

$$j\dagger \equiv \arg\max_j q_f(j) \tag{26}$$

Table 10
The sequential algorithm YS.

(0) Given: $S^{(0)}$, $\theta^{(0)}$, with $\theta^{(0)}_{gj} = 1/L$ for $g \notin S^{(0)}$
(1) Initialization
  (1.1) Set $S = S^{(0)}$, $\theta = \theta^{(0)}$
  (1.2) For each example $x \in X$
    If x possesses a feature in $S^{(0)}$, set $Y_x = \hat{y}$, else set $Y_x = \perp$
(2) Loop:
  (2.1) Choose a feature $f \notin S^{(0)}$ such that $\Lambda_f \neq \emptyset$ and $\theta_f \neq q_f$
    If there is none, stop
  (2.2) Add f to S
  (2.3) For each label j, set $\theta_{fj} = \text{update}(f, j)$
  (2.4) For each example x possessing a feature in S, set $Y_x = \hat{y}$

Note that peaked precision involves a variable amount of "peaking"; the mixing parameters depend on the relative proportions of labeled and unlabeled examples. Note also that $j\dagger$ is a function of f, though we do not explicitly represent that dependence.

The three instantiations of algorithm YS that we consider are

YS-P ("peaked")	$\theta_{fj} = \hat{q}_f(j)$
YS-R ("raw")	$\theta_{fj} = q_f(j)$
YS-FS ("fixed smoothing")	$\theta_{fj} = \tilde{q}_f(j)$

We will show that the first two algorithms reduce K in each iteration. We will show that the third algorithm, YS-FS, reduces K in iterations in which f_t is a new feature, not previously selected. Unfortunately, we are unable to show that YS-FS reduces K when f_t is a previously selected feature. This suggests employing a mixed algorithm in which smoothed precision is used for new features but raw or peaked precision is used for previously selected features.

A final issue with the algorithm schema YS concerns the selection of features in step 2.1. The schema as stated does not specify which feature is to be selected. In essence, the manner in which rules are selected does not matter, as long as one selects rules that have room for improvement, in the sense that the current prediction distribution θ_f differs from raw precision q_f. (The justification for this choice is given in Theorem 9.) The theorems in the following sections show that K decreases in each iteration, so long as any such rule can be found.

One could choose greedily by choosing the feature that maximizes gain G (equation (27)), though in the next section we give lower bounds for G that are rather more easily computed (Theorems 6 and 7).

4.2 Gain

From this point on, we consider a single iteration of the YS algorithm and discard the variable t. We write θ^{old} and ϕ^{old} for the parameter set and labeling at the beginning of the iteration, and we write simply θ and ϕ for the new parameter set and new labeling. The set Λ (respectively, V) represents the examples that are labeled (respectively, unlabeled) at the beginning of the iteration. The selected feature is f.

We wish to choose a prediction distribution for f so as to guarantee that K decreases in each iteration. The **gain** in the current iteration is

$$G = \sum_{x \in X} \sum_{g \in F_x} \left[H(\phi^{\text{old}}_x \| \theta^{\text{old}}_g) - H(\phi_x \| \theta_g) \right] \tag{27}$$

Gain is the negative change in K; it is positive when K decreases.

In considering the reduction in K from $(\phi^{\text{old}}, \theta^{\text{old}})$ to (ϕ, θ), it will be convenient to consider the following intermediate values:

$$\begin{aligned} K_0 &= \sum_{x \in X} \sum_{g \in F_x} H(\phi_x^{\text{old}} \| \theta_g^{\text{old}}) \\ K_1 &= \sum_{x \in X} \sum_{g \in F_x} H(\psi_x \| \theta_g^{\text{old}}) \\ K_2 &= \sum_{x \in X} \sum_{g \in F_x} H(\psi_x \| \theta_g) \\ K_3 &= \sum_{x \in X} \sum_{g \in F_x} H(\phi_x \| \theta_g) \end{aligned}$$

where

$$\psi_{xj} = \begin{cases} [\![j = j*]\!] & \text{if } x \in \mathrm{V}_f \\ \phi_{xj}^{\text{old}} & \text{otherwise} \end{cases}$$

and

$$j* \equiv \arg\max_j \theta_{fj} \tag{28}$$

One should note that

- θ_f is the new prediction distribution for the candidate f; $\theta_{gj} = \theta_{gj}^{\text{old}}$ for $g \neq f$.
- ϕ is the new label distribution, after relabeling. It is defined as

$$\phi_{xj} = \begin{cases} [\![j = \hat{y}(x)]\!] & \text{if } x \in \Lambda \cup X_f \\ \frac{1}{L} & \text{otherwise} \end{cases} \tag{29}$$

- for $x \in \mathrm{V}_f$, the only selected feature at $t+1$ is f, hence $j* = \hat{y}$ for such examples. It follows that ψ and ϕ agree on examples in V_f. They also agree on examples that are unlabeled at $t+1$, assigning them the uniform label distribution. If ψ and ϕ differ, it is only on old labeled examples (Λ) that need to be relabeled, given the addition of f.

The gain G can be represented as the sum of three intermediate gains, corresponding to the intermediate values just defined:

$$G = G_{\mathrm{V}} + G_\theta + G_\Lambda \tag{30}$$

where

$$\begin{aligned} G_{\mathrm{V}} &= K_0 - K_1 \\ G_\theta &= K_1 - K_2 \\ G_\Lambda &= K_2 - K_3 \end{aligned}$$

The gain G_{V} intuitively represents the gain that is attributable to labeling previously unlabeled examples in accordance with the predictions of θ. The gain G_θ represents the gain that is attributable to changing the values θ_{fj}, where f is the selected feature. The

gain G_Λ represents the gain that is attributable to changing the labels of previously labeled examples to make labels agree with the predictions of the new model θ. The gain G_θ corresponds to step 2.3 of algorithm YS, in which θ is changed but ϕ is held constant; and the combined G_V and G_Λ gains correspond to step 2.4 of algorithm YS, in which ϕ is changed while holding θ constant.

In the remainder of this section, we derive two lower bounds for G. In following sections, we show that the updates YS-P, YS-R, and YS-FS guarantee that the lower bounds given below are non-negative, and hence that G is non-negative.

Lemma 3
$G_V = 0$

Proof
We show that K remains unchanged if we substitute ψ for ϕ^{old} in K_0. The only property of ψ that we need is that it agrees with ϕ^{old} on previously labeled examples.

Since $\psi_x = \phi_x^{\text{old}}$ for $x \in \Lambda$, we need only consider examples in V. Since these examples are unlabeled at the beginning of the iteration, none of their features have been selected, hence $\theta_{gj}^{\text{old}} = 1/L$ for all their features g. Hence

$$\begin{aligned} K_1 &= -\sum_{x \in V} \sum_{g \in F_x} \sum_j \psi_{xj} \log \theta_{gj}^{\text{old}} \\ &= -\sum_{x \in V} \sum_{g \in F_x} \left[\sum_j \psi_{xj} \right] \log \frac{1}{L} \\ &= -\sum_{x \in V} \sum_{g \in F_x} \left[\sum_j \phi_{xj}^{\text{old}} \right] \log \frac{1}{L} \\ &= -\sum_{x \in V} \sum_{g \in F_x} \sum_j \phi_{xj}^{\text{old}} \log \theta_{gj}^{\text{old}} \quad = K_0 \end{aligned}$$

(Note that ψ_{xj} is not in general equal to ϕ_{xj}^{old}, but $\sum_j \psi_{xj}$ and $\sum_j \phi_{xj}^{\text{old}}$ both equal 1.) This shows that $K_0 = K_1$, and hence that $G_V = 0$.

Lemma 4
$G_\Lambda \geq 0$.

We must show that relabeling old labeled examples—that is, setting $\phi_x(j) = [\![j = \hat{y}]\!]$ for $x \in \Lambda$—does not increase K. The proof has the same structure as the proof of Theorem 1 and is omitted.

Lemma 5
G_θ is equal to

$$|\Lambda_f| \left[H(q_f \| \theta_f^{\text{old}}) - H(q_f \| \theta_f) \right] + |V_f| \left[\log L - \log \frac{1}{\theta_{fj*}} \right] \tag{31}$$

Proof
By definition, $G_\theta = K_1 - K_2$, and K_1 and K_2 are identical everywhere except on examples

in X_f. Hence

$$G_\theta = \sum_{x \in X_f} \sum_{g \in F_x} \Big[H(\psi_x \| \theta_g^{\text{old}}) - H(\psi_x \| \theta_g) \Big]$$

We divide this sum into three partial sums:

$$\begin{aligned} G_\theta &= A + B + C \qquad (32) \\ A &= \sum_{x \in \Lambda_f} \Big[H(\psi_x \| \theta_f^{\text{old}}) - H(\psi_x \| \theta_f) \Big] \\ B &= \sum_{x \in \mathrm{V}_f} \Big[H(\psi_x \| \theta_f^{\text{old}}) - H(\psi_x \| \theta_f) \Big] \\ C &= \sum_{x \in X_f} \sum_{g \neq f \in F_x} \Big[H(\psi_x \| \theta_g^{\text{old}}) - H(\psi_x \| \theta_g) \Big] \end{aligned}$$

We consider each partial sum separately:

$$\begin{aligned} A &= \sum_{x \in \Lambda_f} \Big[H(\psi_x \| \theta_f^{\text{old}}) - H(\psi_x \| \theta_f) \Big] \\ &= -\sum_{x \in \Lambda_f} \sum_k \psi_{xk} \Big[\log \theta_{fk}^{\text{old}} - \log \theta_{fk} \Big] \\ &= -\sum_{x \in \Lambda_f} \sum_k [\![x \in \Lambda_k]\!] \Big[\log \theta_{fk}^{\text{old}} - \log \theta_{fk} \Big] \\ &= -\sum_k |\Lambda_{fk}| \Big[\log \theta_{fk}^{\text{old}} - \log \theta_{fk} \Big] \\ &= -|\Lambda_f| \sum_k q_f(k) \Big[\log \theta_{fk}^{\text{old}} - \log \theta_{fk} \Big] \\ &= |\Lambda_f| \Big[H(q_f \| \theta_f^{\text{old}}) - H(q_f \| \theta_f) \Big] \qquad (33) \end{aligned}$$

$$\begin{aligned} B &= \sum_{x \in \mathrm{V}_f} \Big[H(\psi_x \| \theta_f^{\text{old}}) - H(\psi_x \| \theta_f) \Big] \\ &= -\sum_{x \in \mathrm{V}_f} \sum_k \psi_{xk} \Big[\log \theta_{fk}^{\text{old}} - \log \theta_{fk} \Big] \\ &= -\sum_{x \in \mathrm{V}_f} \sum_k [\![k = j*]\!] \Big[\log \theta_{fk}^{\text{old}} - \log \theta_{fk} \Big] \\ &= |\mathrm{V}_f| \left[\log \frac{1}{\theta_{fj*}^{\text{old}}} - \log \frac{1}{\theta_{fj*}} \right] \\ &= |\mathrm{V}_f| \left[\log L - \log \frac{1}{\theta_{fj*}} \right] \qquad (34) \end{aligned}$$

The justification for the last step is a bit subtle. If f is a new feature, not previously selected, then $\theta_{fk}^{\text{old}} = 1/L$ for all k, and the substitution is valid. On the other hand, if f is a previously selected feature, then $|\mathrm{V}_f| = 0$, and even though the substitution of

$1/L$ for $\theta_{fj*}^{\text{old}}$ may not be valid, it is innocuous.

$$\begin{aligned} C &= \sum_{x \in X_f} \sum_{g \neq f \in F_x} \left[H(\psi_x \| \theta_g^{\text{old}}) - H(\psi_x \| \theta_g) \right] \\ &= \sum_{x \in X_f} \sum_{g \neq f \in F_x} \left[H(\psi_x \| \theta_g^{\text{old}}) - H(\psi_x \| \theta_g^{\text{old}}) \right] \\ &= 0 \end{aligned} \tag{35}$$

Combining (32), (33), (34), and (35) yields the lemma.

Theorem 6
G is bounded below by (31).

Proof
Combining (30) with Lemmas 3, 4, and 5.

Theorem 7
G is bounded below by

$$|\Lambda_f| \left[H(q_f \| \theta_f^{\text{old}}) - H(q_f \| \theta_f) \right]$$

Proof
The theorem follows immediately from Theorem 6 if we can show that

$$\log L - \log \frac{1}{\theta_{fj*}} \geq 0$$

Observe first that $\log L = H(u)$. (Recall that $u(j) = 1/L$ is the uniform distribution over labels.) By Lemma 1, we know that

$$\begin{aligned} H(u) - \log \frac{1}{\theta_{fj*}} &\geq H(u) - H(\theta_f) \\ &\geq 0 \end{aligned}$$

The latter follows because the uniform distribution maximizes entropy.

Theorem 8
G is bounded below by

$$|\Lambda_f| \left[D(q_f \| \theta_f^{\text{old}}) - D(q_f \| \theta_f) \right]$$

Proof
Immediate from Theorem 7 and the fact that

$$\begin{aligned} H(q_f \| \theta_f^{\text{old}}) - H(q_f \| \theta_f) &= H(q_f) + D(q_f \| \theta_f^{\text{old}}) - H(q_f) - D(q_f \| \theta_f) \\ &= D(q_f \| \theta_f^{\text{old}}) - D(q_f \| \theta_f) \end{aligned}$$

Theorem 9
If $\theta_f^{\text{old}} \neq q_f$, then there is a choice of θ_f that yields strictly positive gain.

Proof
If $\theta_f^{\text{old}} \neq q_f$, then

$$D(q_f \| \theta_f^{\text{old}}) > 0$$

Setting $\theta_f = q_f$ has the result that

$$|\Lambda_f| \left[D(q_f \| \theta_f^{\text{old}}) - D(q_f \| \theta_f) \right] = |\Lambda_f| D(q_f \| \theta_f^{\text{old}}) > 0$$

Hence $G > 0$ by Theorem 8.

4.3 Algorithm YS-P

We now use the results of the previous section to show that the algorithm YS-P is correct in the sense that it reduces K in every iteration.

Theorem 10
In each iteration of algorithm YS-P, K decreases.

Proof
We wish to show that $G > 0$. By Theorem 6, that is true if expression (31) is positive. By Theorem 9, there exist choices for θ_f that make (31) positive, hence in particular, we guarantee $G > 0$ by maximizing (31). We maximize (31) by minimizing

$$|\Lambda_f| H(q_f \| \theta_f) + |\mathrm{V}_f| \log \frac{1}{\theta_{fj*}} \tag{36}$$

Since

$$H(q_f \| \theta_f) = H(q_f) + D(q_f \| \theta_f)$$

we minimize (36) by minimizing

$$|\Lambda_f| D(q_f \| \theta_f) + |\mathrm{V}_f| \log \frac{1}{\theta_{fj*}} \tag{37}$$

Both terms are nonnegative. The first term is zero if $\theta_f = q_f$. The second term is zero for any distribution that concentrates all its mass in a single label $j*$; it is symmetric in all choices of $j*$ and decreases monotonically as θ_{fj*} approaches one. Hence, the minimum of (37) will have $j*$ equal to the mode of q_f, though it may be more peaked than q_f, at the cost of an increase in the first term, but offset by a decrease in the second term.

Recall that $j\dagger = \arg\max_j q_f(j)$. By the reasoning of the previous paragraph, we know that $j\dagger = j*$ at the minimum of (37). Hence we can minimize (37) by minimizing

$$|\Lambda_f| D(q_f \| \theta_f) - |\mathrm{V}_f| \sum_k [\![k = j\dagger]\!] \log \theta_{fk} \tag{38}$$

We compute the gradient:

$$\begin{aligned} &\frac{\partial}{\partial \theta_{fj}} \left[|\Lambda_f| D(q_f \| \theta_f) - |\mathrm{V}_f| \sum_k [\![k = j\dagger]\!] \log \theta_{fk} \right] \\ &= \frac{\partial}{\partial \theta_{fj}} \left[|\Lambda_f| H(q_f \| \theta_f) - |\Lambda_f| H(q_f) - |\mathrm{V}_f| \sum_k [\![k = j\dagger]\!] \log \theta_{fk} \right] \end{aligned}$$

$$
\begin{aligned}
&= \frac{\partial}{\partial\theta_{fj}}|\Lambda_f|H(q_f\|\theta_f) - \frac{\partial}{\partial\theta_{fj}}|\mathrm{V}_f|\sum_k [\![k = j\dagger]\!]\log\theta_{fk} \\
&= -|\Lambda_f|\frac{\partial}{\partial\theta_{fj}}\sum_k q_f(k)\log\theta_{fk} - |\mathrm{V}_f|\frac{\partial}{\partial\theta_{fj}}\sum_k [\![k = j\dagger]\!]\log\theta_{fk} \\
&= -|\Lambda_f|\frac{\partial}{\partial\theta_{fj}}q_f(j)\log\theta_{fj} - |\mathrm{V}_f|\frac{\partial}{\partial\theta_{fj}}[\![j = j\dagger]\!]\log\theta_{fj} \\
&= -|\Lambda_f|q_f(j)\frac{1}{\theta_{fj}} - |\mathrm{V}_f|[\![j = j\dagger]\!]\frac{1}{\theta_{fj}} \qquad (39)
\end{aligned}
$$

As before, the derivative of the constraint $C_f = 0$ is one, and we minimize (38) under the constraint by solving

$$-|\Lambda_f|q_f(j)\frac{1}{\theta_{fj}} - |\mathrm{V}_f|[\![j = j\dagger]\!]\frac{1}{\theta_{fj}} = \lambda$$

$$\theta_{fj} = \left(-|\Lambda_f|q_f(j) - |\mathrm{V}_f|[\![j = j\dagger]\!]\right)/\lambda \qquad (40)$$

Substituting into the constraint gives us

$$\sum_j \left(-|\Lambda_f|q_f(j) - |\mathrm{V}_f|[\![j = j\dagger]\!]\right)/\lambda = 1$$

$$-|\Lambda_f| - |\mathrm{V}_f| = \lambda$$
$$-|X_f| = \lambda$$

Substituting this back into (40) yields:

$$\theta_{fj} = p(\Lambda|f)q_f(j) + p(\mathrm{V}|f)[\![j = j\dagger]\!] \qquad (41)$$

That is, the maximizing solution is peaked precision, which is the update rule for YS-P.

4.4 Algorithm YS-R

We now show that YS-R also decreases K in each iteration. In fact, it has essentially already been proven.

Theorem 11
Algorithm YS-R decreases K in each iteration.

Proof
In the proof of Theorem 9, we showed that the choice

$$\theta_f = q_f$$

yields strictly positive gain. This is the update rule used by YS-R.

4.5 Algorithm YS-FS

The original Yarowsky algorithm YS-0/DL-0 used smoothed precision with fixed ϵ as update rule. We have been unsuccessful at justifying this choice of update rule in general. However, we are able at least to show that it does decrease K when the selected feature is a new feature, not previously selected.

Theorem 12
Algorithm YS-FS has positive gain in each iteration in which the selected feature has not been previously selected.

Proof
By Theorem 7, gain is positive if

$$H(q_f \| \theta_f^{\text{old}}) > H(q_f \| \theta_f) \tag{42}$$

By the assumption that the selected feature f has not been previously selected, θ_f^{old} is the uniform distribution u, and the left-hand side of (42) is equal to $H(q_f \| u)$. It is easy to verify that $H(p \| u) = H(u)$ for any distribution p; hence the left-hand side of (42) is equal to $H(u)$. Further, YS-FS uses smoothed precision as update rule, $\theta_f = \tilde{q}_f$, so (42) can be rewritten as

$$H(u) > H(q_f \| \tilde{q}_f)$$

This condition does not hold trivially, inasmuch as cross entropy, like divergence, is unbounded. But we can show that it holds in this particular case.

We derive an upper bound for $H(q_f \| \tilde{q}_f)$:

$$\begin{aligned}
H(q_f \| \tilde{q}_f) &= -\sum_j q_f(j) \log \tilde{q}_f(j) \\
&= -\sum_j q_f(j) \log \left[\frac{1}{1+L\epsilon} q_f(j) + \frac{L\epsilon}{1+L\epsilon} u(j) \right] \\
&\leq -\sum_j q_f(j) \left[\frac{1}{1+L\epsilon} \log q_f(j) + \frac{L\epsilon}{1+L\epsilon} \log u(j) \right] \\
&= \frac{1}{1+L\epsilon} H(q_f) + \frac{L\epsilon}{1+L\epsilon} H(q_f \| u) \\
&= \frac{1}{1+L\epsilon} H(q_f) + \frac{L\epsilon}{1+L\epsilon} H(u)
\end{aligned} \tag{43}$$

Observe that

$$H(u) > \frac{1}{1+L\epsilon} H(q_f) + \frac{L\epsilon}{1+L\epsilon} H(u) \tag{44}$$

iff

$$\left[1 - \frac{L\epsilon}{1+L\epsilon} \right] H(u) > \frac{1}{1+L\epsilon} H(q_f)$$

iff

$$H(u) > H(q_f)$$

We know that $H(u) \geq H(q_f)$ because the uniform distribution maximizes entropy. We know that the inequality is strict by the following reasoning. Since f is a new feature, $\theta_f^{\text{old}} = u$. Because of the restriction on step 2.1 in algorithm YS, $\theta_f^{\text{old}} \neq q_f$, hence $q_f \neq u$, and $H(u)$ is strictly greater than $H(q_f)$.

Hence (44) is true, and combining (44) with (43), we have shown (42) to be true, proving the theorem.

5. Minimization of Feature Entropy

At the beginning of the article, the co-training algorithm was mentioned as an alternative to the Yarowsky algorithm. There is in fact a connection between co-training and the Yarowsky algorithm. In the original co-training paper (Blum and Mitchell 1998), it was suggested that the algorithm be understood as seeking to maximize agreement on unlabeled data between classifiers trained on two different "views" of the data. Subsequent work (Dasgupta, Littman, and McAllester 2001) has proven a direct connection between classifier error and such cross-view agreement on unlabeled data.

In the current context, there is also justification for pursuing agreement on unlabeled data. However, the Yarowsky algorithm does not assume the existence of two conditionally independent views of the data. Rather, there is a motivation for seeking agreement on unlabeled data between arbitrary pairs of features.

Recall that our original objective function, H, can be expressed as the sum of an entropy term and a divergence term:

$$H = \sum_{x \in X} \left[H(\phi_x) + D(\phi_x \| \pi_x) \right]$$

As $D(\phi_x \| \pi_x)$ becomes small and $H(\pi_x)$ becomes small, $H(\phi_x)$ necessarily also becomes small; hence we can limit H by limiting $H(\pi_x)$ and $D(\phi_x \| \pi_x)$. Intuitively, we wish to reduce the uncertainty of the model's predictions, while also improving the fit between the model's predictions and the known labels.

Let us focus now on the uncertainty of the model's predictions:

$$\begin{aligned}
H(\pi_x) &= -\sum_j \pi_x(j) \log \pi_x(j) \\
&= -\sum_j \pi_x(j) \log \left[\sum_{g \in F_x} \frac{1}{m} \theta_{gj} \right] \\
&\leq -\sum_j \pi_x(j) \sum_{g \in F_x} \frac{1}{m} \log \theta_{gj} \\
&= -\sum_j \left(\sum_{f \in F_x} \frac{1}{m} \theta_{fj} \right) \sum_{g \in F_x} \frac{1}{m} \log \theta_{gj} \\
&= -\frac{1}{m^2} \sum_{f \in F_x} \sum_{g \in F_x} \sum_j \theta_{fj} \log \theta_{gj} \\
&= \frac{1}{m^2} \sum_{f \in F_x} \sum_{g \in F_x} H(\theta_f \| \theta_g) \\
&= \frac{1}{m^2} \sum_{f \in F_x} \sum_{g \in F_x} \left[H(\theta_f) + D(\theta_f \| \theta_g) \right] \\
&= \frac{1}{m} \sum_{f \in F_x} H(\theta_f) + \frac{1}{m^2} \sum_{f \in F_x} \sum_{g \in F_x} D(\theta_f \| \theta_g) \qquad (45)
\end{aligned}$$

In other words, by decreasing the uncertainty of the prediction distributions of individual features and simultaneously increasing the agreement among features (that is, decreasing their pairwise divergence), we decrease an upper bound on $H(\pi_x)$. This

motivates interfeature agreement without recourse to an assumption of independent views.

6. Conclusion

In this article, we have presented a number of variants of the Yarowsky algorithm, and we have shown that they optimize natural objective functions. We considered first the modified generic Yarowsky algorithm Y-1 and showed that it minimizes the objective function H (which is equivalent to maximizing likelihood), provided that its base learner reduces H.

We then considered three families of specific Yarowsky-like algorithms. The Y-1/DL-EM algorithms (Y-1/DL-EM-Λ and Y-1/DL-EM-X) minimize H but have the disadvantage that the DL-EM base learner has no similarity to Yarowsky's original base learner. A much better approximation to Yarowsky's original base learner is provided by DL-1, and the Y-1/DL-1 algorithms (Y-1/DL-1-R and Y-1/DL-1-VS) were shown to minimize the objective function K, an upper bound for H. Finally, the YS algorithms (YS-P, YS-R, and YS-FS) are sequential variants, reminiscent of the Yarowsky-Cautious algorithm of Collins and Singer; we showed that they minimize K.

To the extent that these algorithms capture the essence of the original Yarowsky algorithm, they provide a formal understanding of Yarowsky's approach. Even if they are deemed to diverge too much from the original to cast light on its workings, they at least represent a new family of bootstrapping algorithms with solid mathematical foundations.

References

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

Blum, Avrim and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*, pages 92–100. Morgan Kaufmann, San Francisco.

Collins, Michael and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, College Park, MD, pages 100–110.

Dasgupta, Sanjoy, Michael Littman, and David McAllester. 2001. PAC generalization bounds for co-training. In *Proceedings of Advances in Neural Information Processing Systems 14 (NIPS)*, Vancouver, British Columbia, Canada.

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