Contrastive Estimation: Training Log-Linear Models on Unlabeled Data*

Noah A. Smith and **Jason Eisner**
Department of Computer Science / Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218 USA
{nasmith,jason}@cs.jhu.edu

Abstract

Conditional random fields (Lafferty et al., 2001) are quite effective at sequence labeling tasks like shallow parsing (Sha and Pereira, 2003) and named-entity extraction (McCallum and Li, 2003). CRFs are *log-linear*, allowing the incorporation of arbitrary features into the model. To train on *unlabeled* data, we require *unsupervised* estimation methods for log-linear models; few exist. We describe a novel approach, *contrastive estimation*. We show that the new technique can be intuitively understood as exploiting *implicit negative evidence* and is computationally efficient. Applied to a sequence labeling problem—POS tagging given a tagging dictionary and unlabeled text—contrastive estimation outperforms EM (with the same feature set), is more robust to degradations of the dictionary, and can largely recover by modeling additional features.

1 Introduction

Finding linguistic structure in raw text is not easy. The classical forward-backward and inside-outside algorithms try to guide probabilistic models to discover structure in text, but they tend to get stuck in local maxima (Charniak, 1993). Even when they avoid local maxima (e.g., through clever initialization) they typically deviate from human ideas of what the "right" structure is (Merialdo, 1994).

One strategy is to incorporate domain knowledge into the model's structure. Instead of blind HMMs or PCFGs, one could use models whose features are crafted to pay attention to a range of domain-specific linguistic cues. *Log-linear* models can be so crafted and have already achieved excellent performance when trained on *annotated* data, where they are known as "maximum entropy" models (Ratnaparkhi et al., 1994; Rosenfeld, 1994).

Our goal is to learn log-linear models from *unannotated* data. Since the forward-backward and inside-outside algorithms are instances of Expectation-Maximization (EM) (Dempster et al., 1977), a natural approach is to construct EM algorithms that handle log-linear models. Riezler (1999) did so, then resorted to an approximation because the true objective function was hard to normalize.

Stepping back from EM, we may generally envision parameter estimation for probabilistic modeling as pushing probability mass toward the training examples. We must consider not only where the learner pushes the mass, but also *from where* the mass is *taken*. In this paper, we describe an alternative to EM: *contrastive estimation* (CE), which (unlike EM) explicitly states the source of the probability mass that is to be given to an example.[1]

One reason is to make normalization *efficient*. Indeed, CE generalizes EM and other practical techniques used to train log-linear models, including conditional estimation (for the supervised case) and Riezler's approximation (for the unsupervised case).

The other reason to use CE is to improve *accuracy*. CE offers an additional way to inject domain knowledge into unsupervised learning (Smith and Eisner, 2005). CE hypothesizes that each positive example in training implies a domain-specific set of examples which are (for the most part) degraded (§2). This class of *implicit negative evidence* provides the source of probability mass for the observed example. We discuss the application of CE to log-linear models in §3.

*This work was supported by a Fannie and John Hertz Foundation fellowship to the first author and NSF ITR grant IIS-0313193 to the second author. The views expressed are not necessarily endorsed by the sponsors. The authors also thank three anonymous ACL reviewers for helpful comments, colleagues at JHU CLSP (especially David Smith and Roy Tromble) and Miles Osborne for insightful feedback, and Eric Goldlust and Markus Dreyer for Dyna language support.

[1] Not to be confused with *contrastive divergence* minimization (Hinton, 2003), a technique for training products of experts.

Proceedings of the 43rd Annual Meeting of the ACL, pages 354–362,
Ann Arbor, June 2005.

We are particularly interested in log-linear models over *sequences*, like the conditional random fields (CRFs) of Lafferty et al. (2001) and weighted CFGs (Miyao and Tsujii, 2002). For a given sequence, implicit negative evidence can be represented as a *lattice* derived by finite-state operations (§4). Effectiveness of the approach on POS tagging using unlabeled data is demonstrated (§5). We discuss future work (§6) and conclude (§7).

2 Implicit Negative Evidence

Natural language is a delicate thing. For any plausible sentence, there are many slight perturbations of it that will make it implausible. Consider, for example, the first sentence of this section. Suppose we choose one of its six words at random and remove it; on this example, odds are two to one that the resulting sentence will be ungrammatical. Or, we could randomly choose two adjacent words and transpose them; none of the results are valid conversational English. The learner we describe here takes into account not only the observed positive example, but also a set of similar but deprecated negative examples.

2.1 Learning setting

Let $\vec{x} = \langle x_1, x_2, ... \rangle$, be our observed example sentences, where each $x_i \in \mathcal{X}$, and let $y_i^* \in \mathcal{Y}$ be the unobserved correct hidden structure for x_i (e.g., a POS sequence). We seek a model, parameterized by $\vec{\theta}$, such that the (unknown) correct analysis y_i^* is the best analysis for x_i (under the model). If y_i^* were observed, a variety of training criteria would be available (see Tab. 1), but y_i^* is unknown, so none apply. Typically one turns to the EM algorithm (Dempster et al., 1977), which locally maximizes

$$\prod_i p\left(X = x_i \mid \vec{\theta}\right) = \prod_i \sum_{y \in \mathcal{Y}} p\left(X = x_i, Y = y \mid \vec{\theta}\right) \quad (1)$$

where X is a random variable over sentences and Y a random variable over analyses (notation is often abbreviated, eliminating the random variables). An often-used alternative to EM is a class of so-called Viterbi approximations, which iteratively find the probabilistically-best $\hat{y}$ and then, on each iteration, solve a supervised problem (see Tab. 1).

joint likelihood (JL)	$\prod_i p\left(x_i, y_i^* \mid \vec{\theta}\right)$
conditional likelihood (CL)	$\prod_i p\left(y_i^* \mid x_i, \vec{\theta}\right)$
classification accuracy (Juang and Katagiri, 1992)	$\sum_i \delta(y_i^*, \hat{y}(x_i))$
expected classification accuracy (Klein and Manning, 2002)	$\sum_i p\left(y_i^* \mid x_i, \vec{\theta}\right)$
negated boosting loss (Collins, 2000)	$-\sum_i p\left(y_i^* \mid x_i, \vec{\theta}\right)^{-1}$
margin (Crammer and Singer, 2001)	γ s.t. $\|\vec{\theta}\| \leq 1; \forall i, \forall y \neq y_i^*,$ $\vec{\theta} \cdot (\vec{f}(x_i, y_i^*) - \vec{f}(x_i, y)) \geq \gamma$
expected local accuracy (Altun et al., 2003)	$\prod_i \prod_j p\left(\ell_j(Y) = \ell_j(y_i^*) \mid x_i, \vec{\theta}\right)$

Table 1: Various supervised training criteria. All functions are written so as to be maximized. None of these criteria are available for *unsupervised* estimation because they all depend on the correct label, y^*.

2.2 A new approach: contrastive estimation

Our approach instead maximizes

$$\prod_i p\left(X_i = x_i \mid X_i \in \mathcal{N}(x_i), \vec{\theta}\right) \quad (2)$$

where the "neighborhood" $\mathcal{N}(x_i) \subseteq \mathcal{X}$ is a set of implicit negative examples plus the example x_i itself. As in EM, $p(x_i \mid ..., \vec{\theta})$ is found by marginalizing over hidden variables (Eq. 1). Note that the $x' \in \mathcal{N}(x_i)$ are not treated as hard negative examples; we merely seek to move probability mass from them to the observed x.

The neighborhood of x, $\mathcal{N}(x)$, contains examples that are perturbations of x. We refer to the mapping $\mathcal{N} : \mathcal{X} \rightarrow 2^{\mathcal{X}}$ as the neighborhood function, and the optimization of Eq. 2 as *contrastive estimation* (CE).

CE seeks to move probability mass from the neighborhood of an observed x_i to x_i itself. The learner hypothesizes that good models are those which discriminate an observed example from its neighborhood. Put another way, the learner assumes not only that x_i is good, but that x_i is locally optimal in example space ($\mathcal{X}$), and that alternative, similar examples (from the neighborhood) are inferior. Rather than explain all of the data, the model must only explain (using hidden variables) why the

observed sentence is better than its neighbors. Of course, the validity of this hypothesis will depend on the form of the neighborhood function.

Consider, as a concrete example, learning natural language syntax. In Smith and Eisner (2005), we define a sentence's neighborhood to be a set of slightly-altered sentences that use the same lexemes, as suggested at the start of this section. While their syntax is degraded, the inferred meaning of any of these altered sentences is typically close to the intended meaning, yet the speaker *chose* x and not one of the other $x' \in \mathcal{N}(x)$. Why? Deletions are likely to violate subcategorization requirements, and transpositions are likely to violate word order requirements—both of which have something to do with syntax. x was the most grammatical option that conveyed the speaker's meaning, hence (we hope) roughly the most grammatical option in the neighborhood $\mathcal{N}(x)$, and the syntactic model should make it so.

3 Log-Linear Models

We have not yet specified the form of our probabilistic model, only that it is parameterized by $\vec{\theta} \in \mathbb{R}^n$. Log-linear models, which we will show are a natural fit for CE, assign probability to an (example, label) pair (x, y) according to

$$p\left(x, y \mid \vec{\theta}\right) \stackrel{\text{def}}{=} \frac{1}{Z\left(\vec{\theta}\right)} u\left(x, y \mid \vec{\theta}\right) \tag{3}$$

where the "unnormalized score" $u(x, y \mid \vec{\theta})$ is

$$u\left(x, y \mid \vec{\theta}\right) \stackrel{\text{def}}{=} \exp\left(\vec{\theta} \cdot \vec{f}(x, y)\right) \tag{4}$$

The notation above is defined as follows. $\vec{f} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^n_{\geq 0}$ is a nonnegative vector feature function, and $\vec{\theta} \in \mathbb{R}^n$ are the corresponding feature weights (the model's parameters). Because the features can take any form and need not be orthogonal, log-linear models can capture arbitrary dependencies in the data and cleanly incorporate them into a model.

$Z(\vec{\theta})$ (the partition function) is chosen so that $\sum_{(x,y)} p(x, y \mid \vec{\theta}) = 1$; i.e., $Z(\vec{\theta}) = \sum_{(x,y)} u(x, y \mid \vec{\theta})$. u is typically easy to compute for a given (x, y), but Z may be much harder to compute. All the objective functions in this paper take the form

$$\prod_i \frac{\sum_{(x,y)\in\mathcal{A}_i} p\left(x, y \mid \vec{\theta}\right)}{\sum_{(x,y)\in\mathcal{B}_i} p\left(x, y \mid \vec{\theta}\right)} \tag{5}$$

likelihood criterion	$\mathcal{A}_i$	$\mathcal{B}_i$
joint	$\{(x_i, y_i^*)\}$	$\mathcal{X} \times \mathcal{Y}$
conditional	$\{(x_i, y_i^*)\}$	$\{x_i\} \times \mathcal{Y}$
marginal (*a là* EM)	$\{x_i\} \times \mathcal{Y}$	$\mathcal{X} \times \mathcal{Y}$
contrastive	$\{x_i\} \times \mathcal{Y}$	$\mathcal{N}(x_i) \times \mathcal{Y}$

Table 2: Supervised (upper box) and unsupervised (lower box) estimation with log-linear models in terms of Eq. 5.

where $\mathcal{A}_i \subset \mathcal{B}_i$ (for each i). For log-linear models this is simply

$$\prod_i \frac{\sum_{(x,y)\in\mathcal{A}_i} u\left(x, y \mid \vec{\theta}\right)}{\sum_{(x,y)\in\mathcal{B}_i} u\left(x, y \mid \vec{\theta}\right)} \tag{6}$$

So there is no need to compute $Z(\vec{\theta})$, but we *do* need to compute sums over $\mathcal{A}$ and $\mathcal{B}$. Tab. 2 summarizes some concrete examples; see also §3.1–3.2.

We would prefer to choose an objective function such that these sums are easy. CE focuses on choosing appropriate small contrast sets $\mathcal{B}_i$, both for efficiency and to guide the learner. The natural choice for $\mathcal{A}_i$ (which is usually easier to sum over) is the set of (x, y) that are consistent with what was observed (partially or completely) about the ith training example, i.e., the numerator $\sum_{(x,y)\in\mathcal{A}_i} p(x, y \mid \vec{\theta})$ is designed to find $p(\text{observation } i \mid \vec{\theta})$. The idea is to focus the probability mass within $\mathcal{B}_i$ on the subset $\mathcal{A}_i$ where the i the training example is known to be.

It is possible to build log-linear models where each x_i is a sequence.[2] In this paper, each model is a weighted finite-state automaton (WFSA) where states correspond to POS tags. The parameter vector $\vec{\theta} \in \mathbb{R}^n$ specifies a weight for each of the n transitions in the automaton. y is a hidden path through the automaton (determining a POS sequence), and x is the string it emits. $u(x, y \mid \vec{\theta})$ is defined by applying $\exp$ to the total weight of all transitions in y. This is an example of Eqs. 4 and 6 where $f_j(x, y)$ is the number of times the path y takes the jth transition.

The partition function $Z(\vec{\theta})$ of the WFSA is found by adding up the u-scores of all paths through the WFSA. For a k-state WFSA, this equates to solving a linear system of k equations in k variables (Tarjan, 1981). But if the WFSA contains cycles this infinite sum may diverge. Alternatives to exact com-

[2] These are exemplified by CRFs (Lafferty et al., 2001), which can be viewed alternately as undirected dynamic graphical models with a chain topology, as log-linear models over entire sequences with local features, or as WFSAs. Because "CRF" implies CL estimation, we use the term "WFSA."

putation, like random sampling (see, e.g., Abney, 1997), will not help to avoid this difficulty; in addition, convergence rates are in general unknown and bounds difficult to prove. We would prefer to sum over finitely many paths in $\mathcal{B}_i$.

3.1 Parameter estimation (supervised)

For log-linear models, both CL and JL estimation (Tab. 1) are available. In terms of Eq. 5, both set $\mathcal{A}_i = \{(x_i, y_i^*)\}$. The difference is in $\mathcal{B}$: for JL, $\mathcal{B}_i = \mathcal{X} \times \mathcal{Y}$, so summing over $\mathcal{B}_i$ is equivalent to computing the partition function $Z(\vec{\theta})$. Because that sum is typically difficult, CL is preferred; $\mathcal{B}_i = \{x_i\} \times \mathcal{Y}$ for x_i, which is often tractable. For sequence models like WFSAs it is computed using a dynamic programming algorithm (the forward algorithm for WFSAs). Klein and Manning (2002) argue for CL on grounds of accuracy, but see also Johnson (2001). See Tab. 2; other contrast sets $\mathcal{B}_i$ are also possible.

When $\mathcal{B}_i$ contains only x_i paired with the current best competitor ($\hat{y}$) to y_i^*, we have a technique that resembles maximum margin training (Crammer and Singer, 2001). Note that $\hat{y}$ will then change across training iterations, making $\mathcal{B}_i$ dynamic.

3.2 Parameter estimation (unsupervised)

The difference between supervised and unsupervised learning is that in the latter case, $\mathcal{A}_i$ is forced to sum over label sequences y because they weren't observed. In the unsupervised case, CE maximizes

$$\mathcal{L}_{\mathcal{N}}\left(\vec{\theta}\right) = \log \prod_i \frac{\sum_{y \in \mathcal{Y}} u\left(x_i, y \mid \vec{\theta}\right)}{\sum_{(x,y) \in \mathcal{N}(x_i) \times \mathcal{Y}} u\left(x, y \mid \vec{\theta}\right)} \quad (7)$$

In terms of Eq. 5, $\mathcal{A} = \{x_i\} \times \mathcal{Y}$ and $\mathcal{B} = \mathcal{N}(x_i) \times \mathcal{Y}$. EM's objective function (Eq. 1) is a special case where $\mathcal{N}(x_i) = \mathcal{X}$, for all i, and the denominator becomes $Z(\vec{\theta})$. An alternative is to restrict the neighborhood to the set of observed training examples rather than all possible examples (Riezler, 1999; Johnson et al., 1999; Riezler et al., 2000):

$$\prod_i \left[u\left(x_i \mid \vec{\theta}\right) \Big/ \sum_j u\left(x_j \mid \vec{\theta}\right) \right] \quad (8)$$

Viewed as a CE method, this approach (though effective when there are few hypotheses) seems misguided; the objective says to move mass to each example at the expense of all other training examples.

Another variant is *conditional* EM. Let x_i be a pair $(x_{i,1}, x_{i,2})$ and define the neighborhood to be $\mathcal{N}(x_i) = \{\bar{x} = (\bar{x}_1, x_{i,2})\}$. This approach has been applied to conditional densities (Jebara and Pentland, 1998) and conditional training of acoustic models with hidden variables (Valtchev et al., 1997).

Generally speaking, CE is equivalent to some kind of EM when $\mathcal{N}(\cdot)$ is an equivalence relation on examples, so that the neighborhoods partition $\mathcal{X}$. Then if q is any fixed (untrained) distribution over neighborhoods, CE equates to running EM on the model defined by

$$p'\left(x, y \mid \vec{\theta}\right) \stackrel{\text{def}}{=} q\left(\mathcal{N}(x)\right) \cdot p\left(x, y \mid \mathcal{N}(x), \vec{\theta}\right) \quad (9)$$

CE may also be viewed as an importance sampling approximation to EM, where the sample space $\mathcal{X}$ is replaced by $\mathcal{N}(x_i)$. We will demonstrate experimentally that CE is not just an approximation to EM; it makes sense from a modeling perspective.

In §4, we will describe neighborhoods of sequences that can be represented as acyclic *lattices* built directly from an observed sequence. The sum over $\mathcal{B}_i$ is then the total u-score in our model of all paths in the neighborhood lattice. To compute this, intersect the WFSA and the lattice, obtaining a new *acyclic* WFSA, and sum the u-scores of all its paths (Eisner, 2002) using a simple dynamic programming algorithm akin to the forward algorithm. The sum over $\mathcal{A}_i$ may be computed similarly.

CE with lattice neighborhoods is not confined to the WFSAs of this paper; when estimating weighted CFGs, the key algorithm is the inside algorithm for lattice parsing (Smith and Eisner, 2005).

3.3 Numerical optimization

To maximize the neighborhood likelihood (Eq. 7), we apply a standard numerical optimization method (L-BFGS) that iteratively climbs the function using knowledge of its value and gradient (Liu and Nocedal, 1989). The partial derivative of $\mathcal{L}_{\mathcal{N}}$ with respect to the jth feature weight θ_j is

$$\frac{\partial \mathcal{L}_{\mathcal{N}}}{\partial \theta_j} = \sum_i \mathbf{E}_{\vec{\theta}}\left[f_j \mid x_i\right] - \mathbf{E}_{\vec{\theta}}\left[f_j \mid \mathcal{N}(x_i)\right] \quad (10)$$

This looks similar to the gradient of log-linear likelihood functions on complete data, though the expectation on the left is in those cases replaced by an observed feature value $f_j(x_i, y_i^*)$. In this paper, the

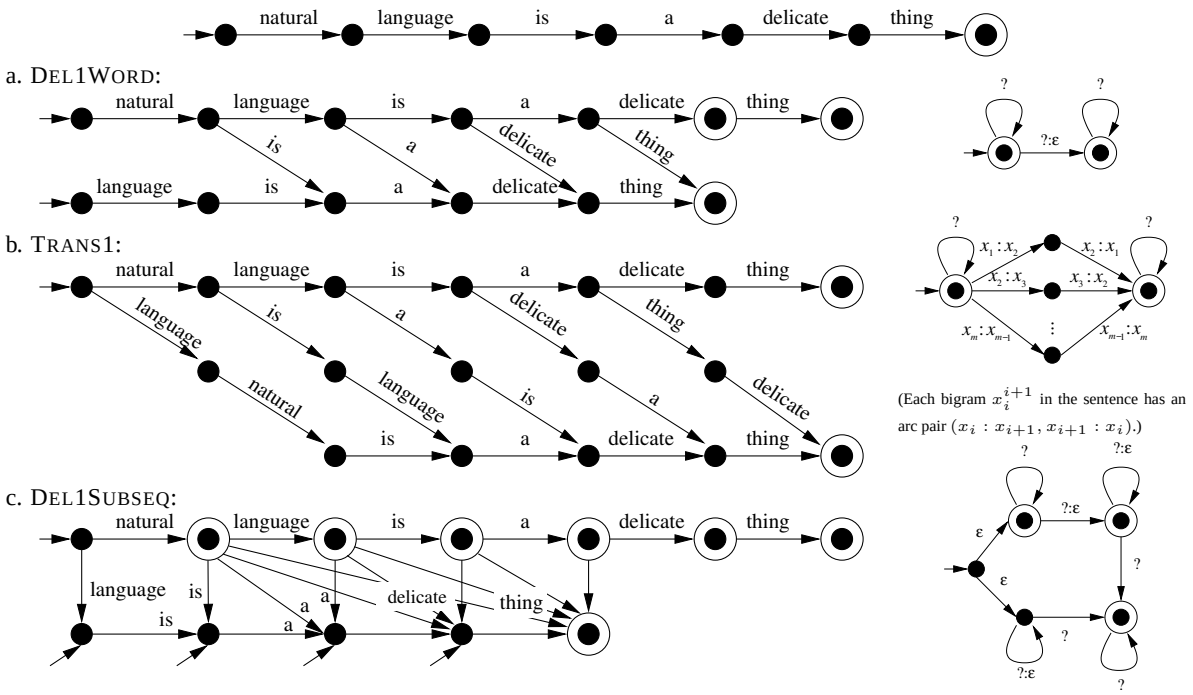


Figure 1: A sentence and three lattices representing some of its neighborhoods. The transducer used to generate each neighborhood lattice (via composition with the sentence, followed by determinization and minimization) is shown to its right.

expectations in Eq. 10 are computed by the forward-backward algorithm generalized to lattices.

We emphasize that the function $\mathcal{L}_{\mathcal{N}}$ is not globally concave; our search will lead only to a local optimum.[3] Therefore, as with all unsupervised statistical learning, the bias in the initialization of $\vec{\theta}$ will affect the quality of the estimate and the performance of the method. In future we might wish to apply techniques for avoiding local optima, such as deterministic annealing (Smith and Eisner, 2004).

4 Lattice Neighborhoods

We next consider some non-classical neighborhood functions for sequences. When $\mathcal{X} = \Sigma^+$ for some symbol alphabet Σ, certain kinds of neighborhoods have natural, compact representations. Given an input string $x = \langle x_1, x_2, ..., x_m \rangle$, we write x_i^j for the substring $\langle x_i, x_{i+1}, ..., x_j \rangle$ and x_1^m for the whole string. Consider first the neighborhood consisting of all sequences generated by deleting a single symbol from the m-length sequence x_1^m:

$$\textsc{Del1Word}(x_1^m) = \left\{ x_1^{\ell-1} x_{\ell+1}^m \mid 1 \leq \ell \leq m \right\} \cup \{x_1^m\}$$

This set consists of $m + 1$ strings and can be compactly represented as a lattice (see Fig. 1a). Another neighborhood involves transposing any pair of adjacent words:

$$\textsc{Trans1}(x_1^m) = \left\{ x_1^{\ell-1} x_{\ell+1} x_\ell x_{\ell+2}^m \mid 1 \leq \ell < m \right\} \cup \{x_1^m\}$$

This set can also be compactly represented as a lattice (Fig. 1b). We can combine DEL1WORD and TRANS1 by taking their union; this gives a larger neighborhood, DELORTRANS1.[4]

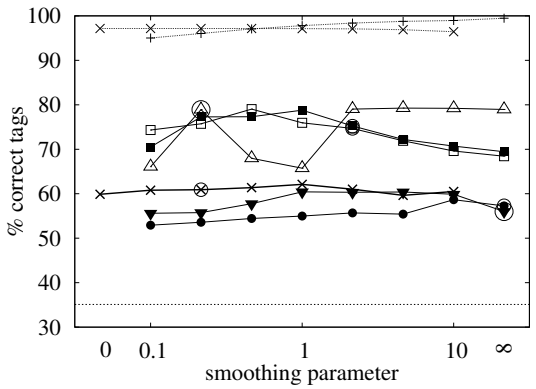
The DEL1SUBSEQ neighborhood allows the deletion of any contiguous subsequence of words that is strictly smaller than the whole sequence. This lattice is similar to that of DEL1WORD, but adds some arcs (Fig. 1c); the size of this neighborhood is $O(m^2)$.

A final neighborhood we will consider is LENGTH, which consists of Σ^m. CE with the LENGTH neighborhood is very similar to EM; it is equivalent to using EM to estimate the parameters of a model defined by Eq. 9 where q is any fixed (untrained) distribution over lengths.

When the vocabulary Σ is the set of words in a natural language, it is never fully known; approximations for defining LENGTH $= \Sigma^m$ include using observed Σ from the training set (as we do) or adding a special OOV symbol.

[3] Without any hidden variables, $\mathcal{L}_{\mathcal{N}}$ *is* globally concave.

[4] In general, the lattices are obtained by composing the observed sequence with a small FST and determinizing and minimizing the result; the relevant transducers are shown in Fig. 1.



	12K		24K		48K		96K	
	sel.	*oracle*	sel.	*oracle*	sel.	*oracle*	sel.	*oracle*
CRF (supervised)		*100.0*		*99.8*		*99.8*		*99.5*
HMM (supervised)		*99.3*		*98.5*		*97.9*		*97.2*
LENGTH	**74.9**	***77.4***	**78.7**	***81.5***	**78.3**	***81.3***	**78.9**	***79.3***
DELORTR1	70.8	*70.8*	78.6	*78.6*	**78.3**	*79.1*	75.2	*78.8*
TRANS1	72.7	*72.7*	77.2	*77.2*	78.1	*79.4*	74.7	*79.0*
EM	49.5	*52.9*	55.5	*58.0*	59.4	*60.9*	60.9	*62.1*
DEL1WORD	55.4	*55.6*	58.6	*60.3*	59.9	*60.2*	59.9	*60.4*
DEL1SSQ	53.0	*53.3*	55.0	*56.7*	55.3	*55.4*	57.3	*58.7*
random expected		*35.2*		*35.1*		*35.1*		*35.1*
ambiguous words		6,244		12,923		25,879		51,521

Figure 2: Percent ambiguous words tagged correctly in the 96K dataset, as the smoothing parameter (λ in the case of EM, σ^2 in the CE cases) varies. The model selected from each criterion using unlabeled development data is circled in the plot. Dataset size is varied in the table at right, which shows models selected using unlabeled development data ("sel.") and using an oracle ("*oracle*," the highest point on a curve). Across conditions, some neighborhood roughly splits the difference between supervised models and EM.

5 Experiments

We compare CE (using neighborhoods from §4) with EM on POS tagging using unlabeled data.

5.1 Comparison with EM

Our experiments are inspired by those in Merialdo (1994); we train a trigram tagger using only unlabeled data, assuming complete knowledge of the tagging dictionary.[5] In our experiments, we varied the amount of data available (12K–96K words of WSJ), the heaviness of smoothing, and the estimation criterion. In all cases, training stopped when the relative change in the criterion fell below 10^{-4} between steps (typically ≤ 100 steps). For this corpus and tag set, on average, a tagger must decide between 2.3 tags for a given token.

The generative model trained by EM was identical to Merialdo's: a second-order HMM. We smoothed using a flat Dirichlet prior with single parameter λ for all distributions (λ-values from 0 to 10 were tested).[6] The model was initialized uniformly.

The log-linear models trained by CE used the same feature set, though the feature weights are no longer log-probabilities and there are no sum-to-one constraints. In addition to an unsmoothed trial, we tried diagonal Gaussian priors (quadratic penalty) with σ^2 ranging from 0.1 to 10. The models were initialized with all $\theta_j = 0$.

Unsupervised model selection. For each (criterion, dataset) pair, we selected the smoothing trial that gave the highest estimation criterion score on a 5K-word development set (also unlabeled).

Results. The plot in Fig. 2 shows the Viterbi accuracy of each criterion trained on the 96K-word dataset as smoothing was varied; the table shows, for each (criterion, dataset) pair the performance of the selected λ or σ^2 and the one chosen by an oracle. LENGTH, TRANS1, and DELORTRANS1 are consistently the best, far out-stripping EM. These gains dwarf the performance of EM on over 1.1M words (66.6% as reported by Smith and Eisner (2004)), even when the latter uses improved search (70.0%). DEL1WORD and DEL1SUBSEQ, on the other hand, are poor, even worse than EM on larger datasets.

An important result is that neighborhoods do not succeed by virtue of *approximating* log-linear EM; if that were so, we would expect larger neighborhoods (like DEL1SUBSEQ) to out-perform smaller ones (like TRANS1)—this is not so. DEL1SUBSEQ and DEL1WORD are poor because they do not give helpful classes of negative evidence: deleting a word or a short subsequence often does very little damage. Put another way, models that do a good job of explaining why no word or subsequence should be deleted do not do so using the familiar POS categories.

The LENGTH neighborhood is as close to log-linear EM as it is practical to get. The inconsistencies in the LENGTH curve (Fig. 2) are notable and also appeared at the other training set sizes. Believing this might be indicative of brittleness in Viterbi label selection, we computed the *expected*

[5] Without a tagging dictionary, tag names are interchangeable and cannot be evaluated on gold-standard accuracy. We address the tagging dictionary assumption in §5.2.

[6] This is equivalent to add-λ smoothing within every M step.

words in tagging dict.	DELORTRANS1 trigram sel.	DELORTRANS1 trigram *oracle*	DELORTRANS1 trigram + spelling sel.	DELORTRANS1 trigram + spelling *oracle*	TRANS1 trigram sel.	TRANS1 trigram *oracle*	TRANS1 trigram + spelling sel.	TRANS1 trigram + spelling *oracle*	LENGTH trigram sel.	LENGTH trigram *oracle*	LENGTH trigram + spelling sel.	LENGTH trigram + spelling *oracle*	EM trigram sel.	EM trigram *oracle*	*random expected*	*ambiguous words*	*ave. tags/token*
all train & dev.	78.3	*90.1*	80.9	*91.1*	**90.4**	*90.4*	88.7	*90.9*	87.8	*90.4*	87.1	***91.9***	78.0	*84.4*	*69.5*	13,150	2.3
1st 500 sents.	72.3	*84.8*	80.2	***90.8***	80.8	*82.9*	**88.1**	*90.1*	68.1	*78.3*	76.9	*83.2*	77.2	*80.5*	*60.5*	13,841	3.7
count ≥ 2	69.5	*81.3*	**79.5**	***90.3***	77.0	*78.6*	78.7	*90.1*	65.3	*75.2*	73.3	*73.8*	70.1	*70.9*	*56.6*	14,780	4.4
count ≥ 3	65.0	*77.2*	78.3	***89.8***	71.7	*73.4*	**78.4**	*89.5*	62.8	*72.3*	73.2	*73.6*	66.5	*66.5*	*51.0*	15,996	5.5

Table 3: Percent of *all* words correctly tagged in the 24K dataset, as the tagging dictionary is diluted. Unsupervised model selection ("sel.") and oracle model selection ("*oracle*") across smoothing parameters are shown. Note that we evaluated on *all* words (unlike Fig. 3) and used 17 coarse tags, giving higher scores than in Fig. 2.

accuracy of the LENGTH models; the same "dips" were present. This could indicate that the learner was trapped in a local maximum, suggesting that, since other criteria did not exhibit this behavior, LENGTH might be a bumpier objective surface. It would be interesting to measure the bumpiness (sensitivity to initial conditions) of different contrastive objectives.[7]

5.2 Removing knowledge, adding features

The assumption that the tagging dictionary is completely known is difficult to justify. While a POS lexicon might be available for a new language, certainly it will not give exhaustive information about all word types in a corpus. We experimented with removing knowledge from the tagging dictionary, thereby increasing the difficulty of the task, to see how well various objective functions could recover. One means to recovery is the addition of features to the model—this is easy with log-linear models but not with classical generative models.

We compared the performance of the best neighborhoods (LENGTH, DELORTRANS1, and TRANS1) from the first experiment, plus EM, using three *diluted* dictionaries and the original one, on the 24K dataset. A diluted dictionary adds (tag, word) entries so that rare words are allowed with *any* tag, simulating zero prior knowledge about the word. "Rare" might be defined in different ways; we used three definitions: words unseen in the first 500 sentences (about half of the 24K training corpus); singletons (words with count ≤ 1); and words with count ≤ 2. To allow more trials, we projected the original 45 tags onto a coarser set of 17 (e.g., RB$*$ $\rightarrow$ADV).

To take better advantage of the power of log-linear models—specifically, their ability to incorporate novel features—we also ran trials augmenting the model with *spelling* features, allowing exploitation of correlations between *parts* of the word and a possible tag. Our spelling features included all observed 1-, 2-, and 3-character suffixes, initial capitalization, containing a hyphen, and containing a digit.

Results. Fig. 3 plots tagging accuracy (on ambiguous words) for each dictionary on the 24K dataset. The x-axis is the smoothing parameter (λ for EM, σ^2 for CE). Note that the different plots are not comparable, because their y-axes are based on different sets of ambiguous words.

So that models under different dilution conditions could be compared, we computed accuracy on *all* words; these are shown in Tab. 3. The reader will notice that there is often a large gap between unsupervised and oracle model selection; this draws attention to a need for better unsupervised regularization and model selection techniques.

Without spelling features, all models perform worse as knowledge is removed. But LENGTH suffers most substantially, relative to its initial performance. Why is this? LENGTH (like EM) requires the model to explain why a given sentence was seen instead of some other sentence of the same length. One way to make this explanation is to manipulate emission weights (i.e., for (tag, word) features): the learner can construct a good class-based *unigram* model of the text (where classes are tags). This is good for the LENGTH objective, but not for learning good POS tag sequences.

In contrast, DELORTRANS1 and TRANS1 do not allow the learner to manipulate emission weights for words not in the sentence. The sentence's goodness must be explained in a way other than by the words it contains: namely through the POS tags. To

[7] A reviewer suggested including a table comparing different criterion values for each learned model (i.e., each neighborhood evaluated on each other neighborhood). This table contained no big surprises; we note only that most models were the best on their own criterion, and among unsupervised models, LENGTH performed best on the CL criterion.

check this intuition, we built local normalized models $p(\text{word} \mid \text{tag})$ from the parameters learned by TRANS1 and LENGTH. For each tag, these were compared by KL divergence to the empirical lexical distributions (from labeled data). For the ten tags accounting for 95.6% of the data, LENGTH more closely matched the empirical lexical distributions. LENGTH is learning a correct distribution, but that distribution is not helpful for the task.

The improvement from adding spelling features is striking: DELORTRANS1 and TRANS1 recover nearly completely (modulo the model selection problem) from the diluted dictionaries. LENGTH sees far less recovery. Hence even our improved feature sets cannot compensate for the choice of neighborhood. This highlights our argument that a neighborhood is not an approximation to log-linear EM; LENGTH tries very hard to approximate log-linear EM but requires a good dictionary to be on par with the other criteria. Good neighborhoods, rather, perform well in their own right.

6 Future Work

Foremost for future work is the "minimally supervised" paradigm in which a small amount of labeled data is available (see, e.g., Clark et al. (2003)). Unlike well-known "bootstrapping" approaches (Yarowsky, 1995), EM and CE have the possible advantage of maintaining posteriors over hidden labels (or structure) throughout learning; bootstrapping either chooses, for each example, a single label, or remains completely agnostic. One can envision a *mixed* objective function that tries to fit the labeled examples while discriminating unlabeled examples from their neighborhoods.[8]

Regardless of how much (if any) data are labeled, the question of good smoothing techniques requires more attention. Here we used a single zero-mean, constant-variance Gaussian prior for all parameters. Better performance might be achieved by allowing different variances for different feature types. This

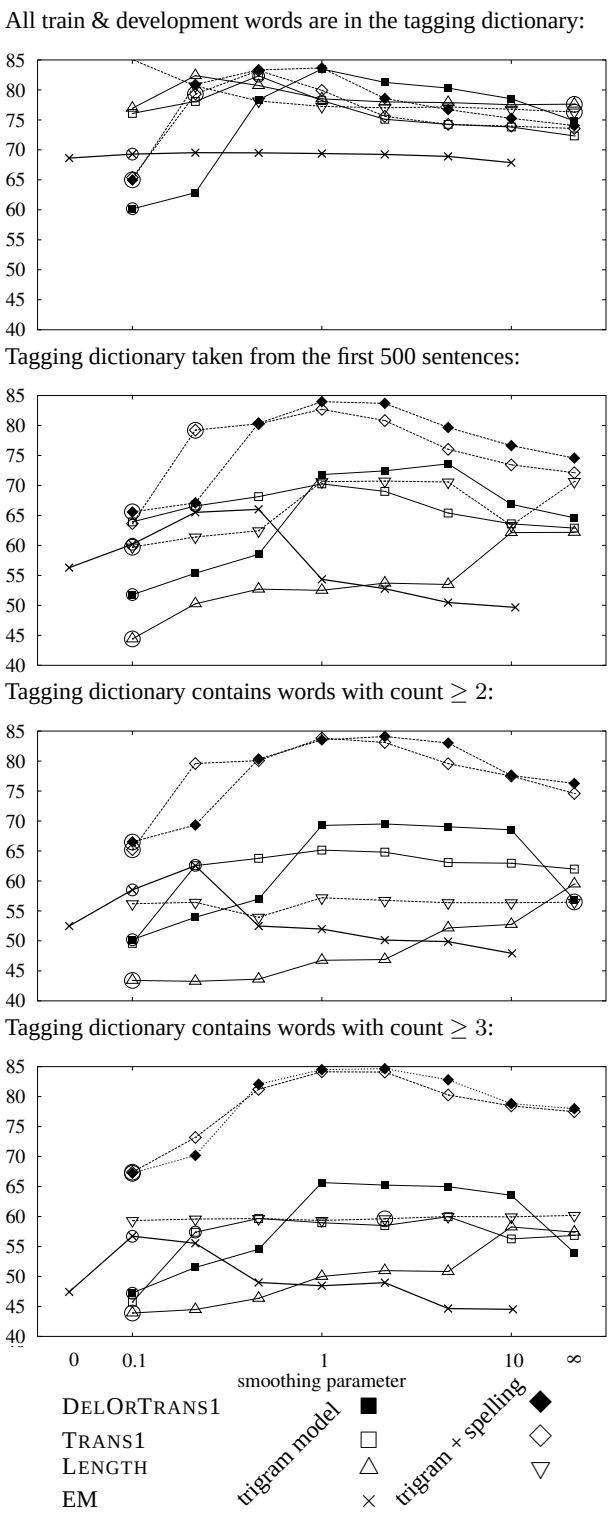


Figure 3: Percent ambiguous words tagged correctly (with coarse tags) on the 24K dataset, as the dictionary is diluted and with spelling features. Each graph corresponds to a different level of dilution. Models selected using unlabeled development data are circled. These plots (unlike Tab. 3) are *not* comparable to each other because each is measured on a different set of ambiguous words.

[8] Zhu and Ghahramani (2002) explored the semi-supervised classification problem for spatially-distributed data, where some data are labeled, using a Boltzmann machine to model the dataset. For them, the Markov random field is over labeling configurations for all examples, not, as in our case, complex structured labels for a particular example. Hence their $\mathcal{B}$ (Eq. 5), though very large, was finite and could be sampled.

leads to a need for more efficient tuning of the prior parameters on development data.

The effectiveness of CE (and different neighborhoods) for dependency grammar induction is explored in Smith and Eisner (2005) with considerable success. We introduce there the notion of designing neighborhoods to guide learning for particular tasks. Instead of guiding an unsupervised learner to match linguists' annotations, the choice of neighborhood might be made to direct the learner toward hidden structure that is helpful for error-correction tasks like spelling correction and punctuation restoration that may benefit from a grammatical model.

Wang et al. (2002) discuss the latent maximum entropy principle. They advocate running EM many times and selecting the local maximum that maximizes entropy. One might do the same for the local maxima of any CE objective, though theoretical and experimental support for this idea remain for future work.

7 Conclusion

We have presented *contrastive estimation*, a new probabilistic estimation criterion that forces a model to explain why the given training data were better than bad data implied by the positive examples. We have shown that for unsupervised sequence modeling, this technique is efficient and drastically outperforms EM; for POS tagging, the gain in accuracy over EM is twice what we would get from ten times as much data and improved search, sticking with EM's criterion (Smith and Eisner, 2004). On this task, with certain neighborhoods, contrastive estimation suffers less than EM does from diminished prior knowledge and is able to exploit new features—that EM can't—to largely recover from the loss of knowledge.

References

S. P. Abney. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4):597–617.

Y. Altun, M. Johnson, and T. Hofmann. 2003. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proc. of EMNLP*.

E. Charniak. 1993. *Statistical Language Learning*. MIT Press.

S. Clark, J. R. Curran, and M. Osborne. 2003. Bootstrapping POS taggers using unlabelled data. In *Proc. of CoNLL*.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. of ICML*.

K. Crammer and Y. Singer. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2(5):265–92.

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

J. Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. of ACL*.

G. E. Hinton. 2003. Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, University College London.

T. Jebara and A. Pentland. 1998. Maximum conditional likelihood via bound maximization and the CEM algorithm. In *Proc. of NIPS*.

M. Johnson, S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proc. of ACL*.

M. Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *Proc. of ACL*.

B.-H. Juang and S. Katagiri. 1992. Discriminative learning for minimum error classification. *IEEE Trans. Signal Processing*, 40:3043–54.

D. Klein and C. D. Manning. 2002. Conditional structure vs. conditional estimation in NLP models. In *Proc. of EMNLP*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

D. C. Liu and J. Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–28.

A. McCallum and W. Li. 2003. Early results for named-entity extraction with conditional random fields. In *Proc. of CoNLL*.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–72.

Y. Miyao and J. Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proc. of HLT*.

A. Ratnaparkhi, S. Roukos, and R. T. Ward. 1994. A maximum entropy model for parsing. In *Proc. of ICSLP*.

S. Riezler, D. Prescher, J. Kuhn, and M. Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proc. of ACL*.

S. Riezler. 1999. *Probabilistic Constraint Logic Programming*. Ph.D. thesis, Universität Tübingen.

R. Rosenfeld. 1994. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Ph.D. thesis, CMU.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*.

N. A. Smith and J. Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *Proc. of ACL*.

N. A. Smith and J. Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *Proc. of IJCAI Workshop on Grammatical Inference Applications*.

R. E. Tarjan. 1981. A unified approach to path problems. *Journal of the ACM*, 28(3):577–93.

V. Valtchev, J. J. Odell, P. C. Woodland, and S. J. Young. 1997. MMIE training of large vocabulary speech recognition systems. *Speech Communication*, 22(4):303–14.

S. Wang, R. Rosenfeld, Y. Zhao, and D. Schuurmans. 2002. The latent maximum entropy principle. In *Proc. of ISIT*.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*.

X. Zhu and Z. Ghahramani. 2002. Towards semi-supervised classification with Markov random fields. Technical Report CMU-CALD-02-106, Carnegie Mellon University.