Bootstrapping Feature-Rich Dependency Parsers with Entropic Priors

David A. Smith and Jason Eisner
Department of Computer Science
Johns Hopkins University
Balitmore, MD 21218, USA
{dasmith,eisner}@jhu.edu

Abstract

One may need to build a statistical parser for a new language, using only a very small labeled treebank together with raw text. We argue that bootstrapping a parser is most promising when the model uses a rich set of redundant features, as in recent models for scoring dependency parses (McDonald et al., 2005). Drawing on Abney's (2004) analysis of the Yarowsky algorithm, we perform bootstrapping by entropy regularization: we maximize a linear combination of conditional likelihood on labeled data and confidence (negative Rényi entropy) on unlabeled data. In initial experiments, this surpassed EM for training a simple feature-poor generative model, and also improved the performance of a feature-rich, conditionally estimated model where EM could not easily have been applied. For our models and training sets, more peaked measures of confidence, measured by Rényi entropy, outperformed smoother ones. We discuss how our feature set could be extended with cross-lingual or cross-domain features, to incorporate knowledge from parallel or comparable corpora during bootstrapping.

1 Motivation

In this paper, we address the problem of bootstrapping new statistical parsers for new languages, genres, or domains.

Why is this problem important? Many applications of multilingual NLP require parsing in order to extract information, opinions, and answers from text, and to produce improved translations. Yet an adequate labeled training corpus—a large **treebank** of manually constructed parse trees of typical sentences—is rarely available and would be prohibitively expensive to develop.

We show how it is possible to train instead from a small hand-labeled treebank in the target domain, together with a large *unannotated* collection of in-domain sentences. Additional resources such as parsers for *other* domains or languages can be integrated naturally.

Dependency parsing is important as a key component in leading systems for information extraction (Weischedel, 2004)[1] and question answering (Peng et al., 2005). These systems rely on edges or paths in dependency parse trees to define their extraction patterns and classification features. Parsing is also key to the latest advances in machine translation, which translate syntactic phrases (Galley et al., 2006; Marcu et al., 2006; Cowan et al., 2006).

2 Our Approach

Our approach rests on three observations:

- Recent "feature-based" parsing models are an excellent fit for bootstrapping, because the parse is often overdetermined by many redundant features.
- The feature-based framework is flexible enough to incorporate other sources of guidance during training or testing—such as the knowledge contained in a parser for another language or domain.
- Maximizing a combination of likelihood on labeled data and confidence on unlabeled data is a principled approach to bootstrapping.

2.1 Feature-Based Parsing

McDonald et al. (2005) introduced a simple, flexible framework for scoring dependency parses. Each directed edge e in the dependency tree is described with a high-dimensional feature vector $\mathbf{f}(e)$. The edge's score is the dot product $\mathbf{f}(e) \cdot \boldsymbol{\theta}$, where $\boldsymbol{\theta}$ is a learned weight vector. The overall score of a dependency tree is the sum of the scores of all edges in the tree.

[1]Ralph Weischedel (p.c.) reports that this system's performance degrades considerably when only phrase chunking is available rather than full parsing.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 667–677, Prague, June 2007.

Given an n-word input sentence, the parser begins by scoring each of the $O(n^2)$ possible edges, and then seeks the highest-scoring legal dependency tree formed by any $n-1$ of these edges, using an $O(n^3)$ dynamic programming algorithm (Eisner, 1996) for projective trees. For non-projective parsing, $O(n^3)$, or with some trickery $O(n^2)$, greedy algorithms exist (Chu and Liu, 1965; Edmonds, 1967; Gabow et al., 1986).

The feature function $\mathbf{f}$ may pay attention to many properties of the directed edge e. Of course, features may consider the parent and child words connected by e, and their parts of speech.[2] But some features used by McDonald et al. (2005) also consider the parts of speech of words *adjacent to* the parent and child, or *between* the parent and child, as well as the *number* of words between the parent and child. In general, these features are not available in a generative model such as a PCFG.

Although feature-based models are often trained purely discriminatively, we will see in §2.6 how to train them to model conditional probabilities.

2.2 Feature-Based Parsing and Bootstrapping

The above parsing model is robust, thanks to its many features. On the Penn Treebank WSJ sections 02–21, for example, McDonald's parser extracts 5.5 million feature types from supervised edges *alone*, with about 120 feature tokens firing per edge. The highest-scoring parse tree represents a *consensus* among all features on all prospective edges. Even if a prospective edge has some discouraging features (i.e., with negative or zero weights), it may still have a relatively high score thanks to its other features. Furthermore, even if the edge has a *low* total score, it may still appear in the consensus parse if the alternatives are even worse or are incompatible with other high-scoring edges.

Put another way, the parser is not able to include high-scoring features or edges independently of one another. Selecting a good feature means accepting all other features on that edge. It also means rejecting various other edges, because of the global constraints that a legal parse tree must give each word only one parent and must be free of cycles and, in the projective case, crossings.

Our observation is that this situation is ideal for so-called "bootstrapping," "co-training," or "minimally supervised" learning methods (Yarowsky, 1995; Blum and Mitchell, 1998; Yarowsky and Wicentowski, 2000). Such methods should thrive when the right answer is overdetermined owing to redundant features and/or global constraints.

Concretely, suppose we start by training a supervised parser on only 100 examples, using some regularization method to prevent overfitting to this set. While many features might truly be relevant to the task, only a few appear often enough in this small training set to acquire significantly positive or negative weights.

Even this lightly trained parser may be quite sure of itself on *some* test sentences in a large unannotated corpus, when one parse scores far higher than all others. More generally, the parser may be sure about *part* of a sentence: it may be certain that a particular edge is present (or absent), because that edge tends to be present (or absent) in all high-scoring parses.

Retraining the feature weights $\boldsymbol{\theta}$ on these high-confidence edges can learn about additional features that are correlated with an edge's success or failure. For example, it may now learn strong weights for lexically specific features that were never observed in the supervised training set. The retrained parser may now be able to confidently parse even more of the unannotated examples; so we can iterate the process.

Our hope is that the model identifies new good and bad edges at each step, and does so correctly. The *more* features and global constraints the model has,

- the more power it will have to discriminate among edges even when $\boldsymbol{\theta}$ is *insufficiently* trained. (Some feature weights may be too weak (i.e., too close to zero) because the initial labeled set is small.)

- the more robust it will be against errors even when $\boldsymbol{\theta}$ is *incorrectly* trained. (Some feature weights may be too strong or have the wrong sign, because of overfitting or mistaken parses during bootstrapping.)

[2] Note that since we are not trying to *predict* parts of speech, we treat the output of one or more automatic taggers as yet more inputs to edge feature functions.

In the former case, strong features lend their strength to weak ones. In the latter case, a conflict among strong features weakens the ones that depart from the consensus, or discounts the example sentence if there is no consensus.

Previous work on parser bootstrapping has not been able to exploit this redundancy among features, because it has used PCFG-like models with far fewer features (Steedman et al., 2003).

2.3 Adaptation and Projection via Features

The previous section assumed that we had a small supervised treebank in the target language and domain (plus a large unsupervised corpus). We now consider other, more dubious, knowledge sources that might supplement or replace this small treebank. In each case, we can use these knowledge sources to derive *features* that may—or may not—prove trustworthy during bootstrapping.

Parses from a different domain. One might have a treebank for a *different* domain or genre of the target language.

One could simply include these trees in the initial supervised training, and hope that bootstrapping corrects any learned weights that are inappropriate to the target domain, as discussed above. In fact, McClosky et al. (2006) found a similar technique to be effective—though only in a model with a large feature space ("PCFG + reranking"), as we would predict.

However, another approach is to train a separate out-of-domain parser, and use this to generate *additional features* on the supervised and unsupervised in-domain data (Blitzer et al., 2006). Bootstrapping now teaches us where to trust the out-of-domain parser. If our basic model has 100 features, we could add features 101 through 200, where for example $f_{123}(e) = f_{23} \cdot \log \tilde{\Pr}(e)$ and $\tilde{\Pr}(e)$ is the posterior edge probability according to the out-of-domain parser. Learning that this feature has a high weight means learning to trust the out-of-domain parser's *decision* on edges where in-domain feature 23 fires. Even more sensibly, we could add features such as $f_{201}(e) = \sum_{i=1}^{10} \tilde{f}_i(e) \cdot \tilde{\theta}_i$, where $\tilde{\mathbf{f}}$ and $\tilde{\boldsymbol{\theta}}$ are the feature and weight vectors for the out-of-domain parser. Learning that this feature has a high weight means learning to trust the out-of-domain parser's *feature weights* for a particular *class* of features (those numbered 1 through 10). This addresses the intuition that some linguistic phenomena remain stable across domains.

Parses of translations. Suppose we have translations into English of *some* of our supervised or unsupervised sentences. Good probabilistic dependency parsers already exist for English, so we run one over the English translation. We can now derive many additional features on candidate edges on the target sentence. For example, dependency edges in the target language of the form $c \xrightarrow{poss} p$ (this denotes a child-to-parent dependency with label *possessor*) might often correspond to dependency paths in the English translation of the form $p' \xleftarrow{prep} \text{of} \xleftarrow{pobj} c'$. To discover whether this is so, we define a feature i by

$$f_i(c \xrightarrow{poss} p) \stackrel{\text{def}}{=} \log \sum_{c',p'} \begin{array}{l} (\Pr(c \text{ aligns with } c') \\ \cdot \Pr(p \text{ aligns with } p') \\ \cdot \Pr(p' \xleftarrow{prep} \text{of} \xleftarrow{pobj} c')) \end{array} \quad (1)$$

where c', p' range over word tokens in the English translation, "of" is a literal English word, and the probabilities are posteriors provided by a probabilistic aligner and a probabilistic English parser. Note that this is a *single* feature (not a feature family parameterized by c, p). It scores any candidate edge on whether it is a $\xrightarrow{poss}$ edge that seems to align to an English $\xleftarrow{prep} \text{of} \xleftarrow{pobj}$ path.

This method is inspired by Hwa et al. (2005), who bootstrapped parsers for Spanish and Chinese by projecting dependencies from English translations and training a new parser on the resulting noisy treebank. They used only 1-best translations, 1-best alignments, dependency paths of length 1, and no labeled data in Spanish or Chinese.

Hwa et al. (2005) used a manually written postprocessor to correct some of the many incorrect projections. By contrast, our framework uses the projected dependencies only as one source of features. They may be overridden by other features in particular cases, and will be given a high weight only if they tend to agree with other features during bootstrapping. A similar *soft* projection of dependencies was used in supervised machine translation by Smith and Eisner (2006), who used a source sentence's dependency paths to *bias* the generation of its translation.

Note that these bilingual features will only fire on those supervised or unsupervised sentences for which we have an English translation. In particular, they will usually be unavailable on the test set. However, we hope that they will seed and facilitate the bootstrapping process, by helping us confidently parse some unsupervised sentences that we would not be able to confidently parse without an English translation.

Parses of comparable English sentences. World knowledge can be useful in parsing. Suppose you see a French sentence that contains *mangeons* and *pommes*, and you know that *manger=eat* and *pomme=apple*. You might reasonably guess that *pommes* is the direct object of *mangeons*, because you know that *apple* is a plausible direct object for *eat*. We can discover this last bit of world knowledge from comparable English text. Translation dictionaries can themselves be induced from comparable corpora (Schafer and Yarowsky, 2002; Schafer, 2006; Klementiev and Roth, 2006), or extracted from bitext or digitized versions of human-readable dictionaries if these are available.

The above inference pattern can be captured by features similar to those in equation (1). For example, one can define a feature j by

$$f_i(c \xrightarrow{poss} p) \stackrel{\text{def}}{=} \log \Pr(p' \xleftarrow{prep} \text{of} \xleftarrow{pobj} c' \mid p' \text{ translates } p, c' \text{ translates } c) \quad (2)$$

where each event in the event space is a pair (c', p') of same-sentence tokens in comparable English text, all pairs being equally likely. Thus, to estimate $\Pr(\cdot \mid \cdot)$, the denominator counts same-sentence token pairs (c', p') in the comparable English corpus that translate into the types (c, p), and the numerator counts such pairs that are *also* related by a $\xleftarrow{prep}$ of $\xleftarrow{pobj}$ path. Since the lexical translations and dependency paths are typically not labeled in the English corpus, a given pair must be counted fractionally according to its posterior probability of satisfying these conditions, given models of contextual translation and English parsing.[3]

[3]Similarly, Jansche (2005) imputes "missing" trees by using comparable corpora.

2.4 Bootstrapping as Optimization

Section 2.2 assumed a relatively conventional kind of bootstrapping, where each iteration retrains the model on the examples where it is currently most confident. This kind of "confidence thresholding" has been popular in previous bootstrapping work (as cited in §2.2). It attempts to maintain high accuracy while gradually expanding coverage. The assumption is that throughout the training procedure, the parser's confidence is a trustworthy guide to its correctness. Different bootstrapping procedures use different learners, smoothing methods, confidence measures, and procedures for "forgetting" the labelings from previous iterations.

In his analysis of Yarowsky (1995), Abney (2004) formulates several variants of bootstrapping. These are shown to increase either the likelihood of the training data, or a lower bound on that likelihood. In particular, Abney defines a function K that is an upper bound on the *negative* log-likelihood, and shows his bootstrapping algorithms locally minimize K.

We now present a generalization of Abney's K function and relate it to another semi-supervised learning technique, entropy regularization (Brand, 1999; Grandvalet and Bengio, 2005; Jiao et al., 2006). Our experiments will tune the feature weight vector, $\boldsymbol{\theta}$, to minimize our function. We will do so simply by applying a *generic* function minimization method (stochastic gradient descent), rather than by crafting a new Yarowsky-style or Abney-style iterative procedure for our specific function.

Suppose we have examples x_i and corresponding possible labelings $y_{i,k}$. We are trying to learn a parametric model $p_{\boldsymbol{\theta}}(y_{i,k} \mid x_i)$. If $\tilde{p}(y_{i,k} \mid x_i)$ is a "labeling distribution" that reflects our uncertainty about the true labels, then our *expected* negative log-likelihood of the model is

$$\begin{aligned} K &\stackrel{\text{def}}{=} -\sum_i \sum_k \tilde{p}(y_{i,k} \mid x_i) \log p_{\boldsymbol{\theta}}(y_{i,k} \mid x_i) \\ &= \sum_i \sum_k \tilde{p}(y_{i,k}|x_i) \log \frac{\tilde{p}(y_{i,k}|x_i)}{p_{\boldsymbol{\theta}}(y_{i,k}|x_i)\tilde{p}(y_{i,k}|x_i)} \\ &= \sum_i D(\tilde{p}_i \| p_{\boldsymbol{\theta},i}) + H(\tilde{p}_i) \quad (3) \end{aligned}$$

where $\tilde{p}_i(\cdot) \stackrel{\text{def}}{=} \tilde{p}(\cdot \mid x_i)$ and $p_{\boldsymbol{\theta},i}(\cdot) \stackrel{\text{def}}{=} p_{\boldsymbol{\theta}}(\cdot \mid x_i)$.

Note that K is a function not only of $\boldsymbol{\theta}$ but also of the labeling distribution $\tilde{p}$; a learner might be allowed to manipulate either in order to decrease K.

The summands of K in equation (3) can be divided into two cases, according to whether x_i is labeled or not. For the labeled examples $\{x_i : i \in L\}$, the labeling distribution $\tilde{p}_i$ is a point distribution that assigns all probability to the true, known label y_i^*. Then $H(\tilde{p}_i) = 0$. The total contribution of these examples to K simplifies to $\sum_{i \in L} -\log p_{\boldsymbol{\theta}}(y_i^* \mid x_i)$, i.e., just the negative log-likelihood on the labeled data.

But what is the labeling distribution for the unlabeled examples $\{x_i : i \notin L\}$? Abney simply uses a uniform distribution over labels (e.g., parses), to reflect that the label is unknown. If his bootstrapping algorithm "labels" x_i, then i moves into L and $H(\tilde{p}_i)$ is thereby reduced from maximal to 0. As a result, a method that labels the most confident examples may reduce K, and Abney shows that his method does so.

Our approach is different: we will take the labeling distribution $\tilde{p}_i$ to be our *actual* current belief $p_{\boldsymbol{\theta},i}$, and manipulate it through changing $\boldsymbol{\theta}$ rather than L. L remains the original set of *supervised* examples. The total contribution of the *unsupervised* examples to K then simplifies to $\sum_{i \notin L} H(p_{\boldsymbol{\theta},i})$.

We have no reason to believe that these two contributions (supervised and unsupervised) should be weighted equally. We thus introduce a multiplier γ to form the actual objective function that we minimize with respect to $\boldsymbol{\theta}$:[4]

$$-\sum_{i \in L} \log p_{\boldsymbol{\theta},i}(y_i^*) + \gamma \sum_{i \notin L}^{N} H(p_{\boldsymbol{\theta},i}) \tag{4}$$

One may regard γ as a Lagrange multiplier that is used to constrain the classifier's uncertainty H to be low, as presented in the work on entropy regularization (Brand, 1999; Grandvalet and Bengio, 2005; Jiao et al., 2006).

Conventional bootstrapping retrains on the *most* confident unsupervised examples, making them more confident. Gradient descent on equation (4) essentially does the same, since unsupervised examples contribute to (4) only through H, and the shape of the H function means that it is most rapidly decreased by making the *most* confident unsupervised examples more confident.

Besides favoring models that are self-confident on the unlabeled data, the objective function (4) also explicitly asks the model to continue to get the correct answers on the initial supervised corpus. $1/\gamma$ controls the strength of this request. One could obtain a similar effect in conventional bootstrapping by upweighting the initial labeled corpus when retraining.

2.5 Online Learning

Minimizing equation (4) for parsing is more computationally intensive than in many other applications of bootstrapping, such as word sense disambiguation or document classification. With millions of features, our objective could take many iterations to converge to a local optimum, if we were only to update our parameter vector $\boldsymbol{\theta}$ after each iteration through a large unsupervised corpus.

For many machine learning problems over large datasets, online learning methods such as **stochastic gradient descent** (SGD) have been empirically observed to converge in fewer iterations (Bottou, 2003). In SGD, instead of taking an optimization step in the direction of the gradient calculated over all unsupervised training examples, we parse each example, calculate the gradient of the objective function evaluated on that example alone, and then take a small step downhill. The update rule is thus

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \cdot \nabla F^{(t)}(\boldsymbol{\theta}^{(t)}) \tag{5}$$

where $\boldsymbol{\theta}^{(t)}$ is the parameter vector at time t, $F^{(t)}(\boldsymbol{\theta})$ is the objective function specialized to the time-t example, and $\eta > 0$ is a learning rate that we choose. We check for convergence after each pass through the example set.

2.6 Algorithms and Complexity

To evaluate equation (4), we need a conditional model of trees given a sentence x_i. We define one by exponentiating and normalizing the tree scores: $p_{\boldsymbol{\theta},i}(y_{i,k}) \stackrel{\text{def}}{=} \exp(\sum_{e \in y_{i,k}} \mathbf{f}(e) \cdot \boldsymbol{\theta})/Z_i$.

With exponentially many parses of x_i, does our objective function (4) now have prohibitive com-

[4] This function is not necessarily convex in $\boldsymbol{\theta}$, because of the addition of the entropy term (Jiao et al., 2006). One might try an annealing strategy: start γ at zero (where the function *is* convex) and gradually increase it, hoping to "ride" the global maximum. Although we could increase γ until the entropy term dominates the minimizations and we approach a completely deterministic classifier, it is preferable to use some labeled heldout data to evaluate a stopping criterion.

putational complexity? The complexity is actually similar to that of the inside algorithm for parsing. In fact, the first term of (4) for projective parsing is found by running the $O(n^3)$ inside algorithm on supervised data,[5] and its gradient is found by the corresponding $O(n^3)$ outside algorithm. For non-projective parsing, the analogy to the inside algorithm is the $O(n^3)$ "matrix-tree algorithm," which is dominated asymptotically by a matrix determinant (Smith and Smith, 2007; Koo et al., 2007; McDonald and Satta, 2007). The gradient of a determinant may be computed by matrix inversion, so evaluating the gradient again has the same $O(n^3)$ complexity as evaluating the function.

The second term of (4) is the Shannon entropy of the posterior distribution over parses. Computing this for projective parsing takes $O(n^3)$ time, using a dynamic programming algorithm that is closely related to the inside algorithm (Hwa, 2000).[6] For non-projective parsing, unfortunately, the runtime rises to $O(n^4)$, since it requires determinants of n distinct matrices (each incorporating a log factor in a different column; we omit the details). The gradient evaluation in both cases is again about as expensive as the function evaluation.

A convenient speedup is to replace Shannon entropy with Rényi entropy. The family of **Rényi entropy** measures is parameterized by α:

$$R_\alpha(p) \quad = \quad \frac{1}{1-\alpha} \log \left(\sum_y p(y)^\alpha \right) \qquad (6)$$

In our setting, where $p = p_{\boldsymbol{\theta},i}$, the events y are the possible parses $y_{i,k}$ of x_i. Observe that under our definition of p, $\sum_y p(y)^\alpha = \{\sum_y \exp[\sum_{e \in y} \mathbf{f}(e) \cdot (\alpha\boldsymbol{\theta})]\}/Z_i^\alpha$. We already have Z_i from running the inside algorithm, and we can find the numerator by running the inside algorithm again with $\boldsymbol{\theta}$ scaled by α. Thus with Rényi entropy, all computations and their gradients are $O(n^3)$—even in the non-projective case.

Rényi entropy is also a theoretically attractive generalization. It can be shown that $\lim_{\alpha \to 1} R_\alpha(p)$ is in fact the Shannon entropy $H(p)$ and that $\lim_{\alpha \to \infty} R_\alpha(p) = -\log \max_y p(y)$, i.e. the negative log probability of the modal or "Viterbi" label (Arndt, 2001; Karakos et al., 2007). The $\alpha = 2$ case, widely used as a measure of purity in decision tree learning, is often called the "Gini index." Finally, when $\alpha = 0$, we get the log of the number of labels, which equals the H(uniform distribution) that Abney used in equation (3).

[5]The numerator of $p_{\boldsymbol{\theta},i}(y_i^*)$ (see definition above) is trivial since y_i^* is a single known parse. But the denominator Z_i is a normalizing constant that sums over all parses; it is found by a dependency-parsing variant of the inside algorithm, following (Eisner, 1996).

[6]See also (Mann and McCallum, 2007) for similar results on conditional random fields.

3 Evaluation

For this paper, we performed some initial bootstrapping experiments on small corpora, using the features from (McDonald et al., 2005). After discussing experimental setup (§3.1), we look at the correlation of confidence with accuracy and with oracle likelihood, and at the fine-grained behaviour of models' dependency edge posteriors (§3.2). We then compare our confidence-maximizing bootstrapping to EM, which has been widely used in semi-supervised learning (§3.4). Section 3.3 presents overall bootstrapping accuracy.

3.1 Experimental Design

We bootstrapped non-projective parsers for languages assembled for the CoNLL dependency parsing competitions (Buchholz and Marsi, 2006). We selected German, Spanish, and Czech (Brants et al., 2002; Civit Torruella and Martí Antonín, 2002; Böhmová et al., 2003). After removing sentences more than 60 words long, we randomly divided each corpus into small seed sets of 100 and 1000 trees; development and test sets of 200 trees each; and an unlabeled training set from the rest.

These treebanks contain strict dependency trees, in the sense that their only nodes are the words and a distinguished root node. In the Czech dataset, more than one word can attach to the root; also, the trees in German, Spanish, and Czech may be non-projective. We use the `MSTParser` implementation described in McDonald et al. (2005) for feature extraction. Since our seed sets are so small, we extracted features from all edges in both the seed and the unlabeled parts of our training data, not just the edges annotated as correct. Since this produced many more features, we pruned our features to those with at least 10 occurrences over all edges.

Rényi α	Correlation of 100-tree model Acc.	Xent.	1000-tree model Acc.	Xent.
(uniform, Abney) 0	-0.254	0.980	-0.180	0.937
.5	-0.256	0.981	-0.203	0.955
(Shannon) 1	-0.260	0.983	-0.220	0.964
(Gini) 2	-0.266	0.985	-0.250	0.977
5	-0.291	0.992	-0.304	0.990
7	-0.301	0.993	-0.341	0.991
(Viterbi) ∞	-0.317	0.995	-0.326	0.992
Xent.	-0.391	1.000	-0.410	1.000

Table 1: Correlation, on development sentences, of Rényi entropy with model accuracy and with cross-entropy ("Xent."). Since these are measures of uncertainty, we see a negative correlation. As α increases, we place more confidence in high-probability parses and correlate better with accuracy.

We used stochastic gradient descent first to minimize equation (4) on the labeled seed sets. Then we continued to optimize over the labeled and unlabeled data together. We tested for convergence using accuracy on development data.

3.2 Empirically Evaluating Entropy

Bootstrapping assumes that where the parser is confident, it tends to be correct. Standard bootstrapping methods retrain directly on confident links; similarly, our approach tries to make the parser even more confident on those links.

Is this assumption really true empirically? Yes: not only does confidence on unlabeled data correlate with cross-entropy, but both confidence and cross-entropy correlate well with accuracy. As we will see, some confidence measures correlate better than others. In particular, measures that are more peaked around the one-best prediction of the parser, as in Viterbi re-estimation, perform well.

If we train a non-projective German parser on small seed sets of 100 and 1000 trees, only, how well does its own confidence predict its performance? For 200 points—labeled development sentences—we measured the linear correlation of various Rényi entropies (6), normalized by sentence length, with tree accuracy (Table 1). We also measured how these normalized Rényi entropies correlate with the posterior log-probability the model assigns to the true parse (the cross-entropy).

Since Rényi entropy is a measure of uncertainty, we see a negative correlation with accuracy. This correlation strengthens as we raise α to ∞, so we might expect Viterbi re-estimation, or a differentiable objective function with a very high α, to perform best on held-out data. Note also that the cross-entropy, which looks at the true labels on the held-out data, does not itself correlate very much better with accuracy than the best *unsupervised* confidence measures. Finally, we see that Rényi entropies with higher α are more stable: when calculated for a model trained on more data, they improve their correlation with accuracy.

From tree confidence, we now turn to edge confidence: what is the posterior probability that a model assigns to each of the n^2 edges in the dependency graph? Figure 1 shows smoothed histograms of true edges (black) and false edges (gray) in held-out data, according to the posterior probabilities we assign to

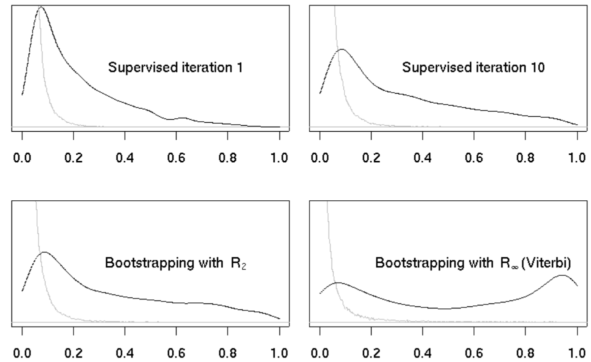

Figure 1: Posterior probability of correct and incorrect edges in German test data under various models. We show the distribution of posterior probabilities for correct edges, known from an oracle, in black and incorrect edges in gray. In the upper row, learning on an initial supervised set raises the posterior probability of correct edges while dragging along some incorrect edges. In the lower row, we see that adding unlabeled data with R_2 entropy continues the pattern of the supervised learner. R_∞ (Viterbi) training induces a second mode in correct posterior probabilities near 1 although it does shift more incorrect edges closer to 1.

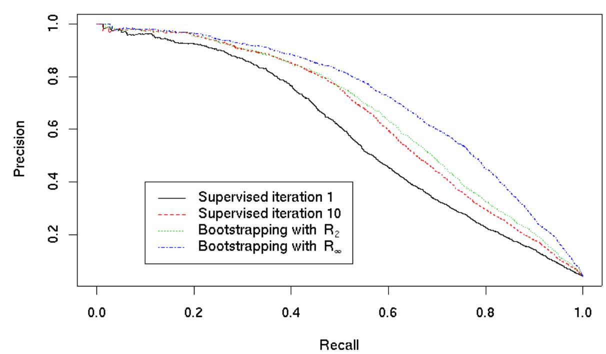

Figure 2: Precision-recall curves for selecting edges according to their posterior probabilities: better bootstrapping puts more area under the curve.

them. Since there are many more false edges, the figures are cropped to zoom in on the distribution of true edges. As we start training on the labeled seed set, the posterior probabilities of true edges move towards one; many false edges also get greater mass, but not to the same extent. As we add unlabeled data, we can see the different learning strategies of different confidence measures. R_2 gradually moves a few true and many fewer false edges towards 1, while R_∞ (Viterbi) learning is so confident as to induce a bimodal distribution in the posteriors of true edges. Figure 2 visualizes the same data as four precision-recall curves, which show how noisy the highest-confidence edges are, across a range of confidence thresholds. Although the very high precision end stays stable after 10 iterations on the seed set, the addition of unlabeled data puts more area under the curve. Again, R_∞ dominates R_2.

3.3 Bootstrapping Results

We performed bootstrapping experiments on the full CoNLL sets for Czech, German, and Spanish using the non-projective model from McDonald et al. (2005). Performance confirms the results of our analysis above (Table 2). Adding unlabeled data improves performance over that of the seed set, with the exception of the Czech data with R_2 bootstrapping. As we saw in §3.2, bootstrapping with R_∞ dominates bootstrapping with R_2 confidence. For comparison, we also show the results obtained by supervised training on the combined seed and unlabeled sets. Recall that we did not use the tree annotations to perform feature selection; models trained with only supported features ought to perform better.

Although we see statistically significant improvements (at the .05 level on a paired permutation test), the quality of the parsers is still quite poor, in contrast to other applications of bootstrapping which "rival supervised methods" (Yarowsky, 1995). Almost certainly, the CoNLL datasets, comprising at most some tens of thousands of sentences per language, are too small to afford qualitative improvements. Also, at these relatively small training sizes, our preliminary attempts to leverage comparable English corpora did not improve performance.

What features were learned, and how dependent is performance on the seed set? We analyzed the performance of German bootstrapping on a development set (Table 3). Using the parameters at the last iteration of supervised training on the seed set as a baseline, we tried updating to their bootstrapped values the weights of only those features that occurred in the seed set. This achieved nearly the same accuracy as updating all the features. As one would expect, using only the non-seed features' weights performs abysmally. This might be the case simply because the seed set is likely to contain frequently occurring features. If, however, we use only the features occurring in an alternate training set of the same size (100 sentences), we get much worse performance. These results indicate that our bootstrapped parser is still heavily dependent on the features that happened to fire in the seed set; we have not "forgotten" our initial conditions. Similar experiments show that unlexicalized features contribute the most to bootstrapping performance. Since in our log-linear models features have been trained to work together, we must not put too much weight on these ablation results. These experiments do, however, suggest that bootstrapping improved our results by refining the values of known, non-lexicalized features.

		% accuracy		
	Seed trees	$\alpha = 0$	2	∞
Czech	100	56.1	54.8	**58.3**
	1000	68.1	68.2	68.2
	71468	77.9	–	–
German	100	60.9	**62.4**	**65.3**
	1000	74.6	74.5	**75.0**
	37745	86.0	–	–
Spanish	100	63.6	**64.1**	**64.4**
	2786	76.6	–	–

Table 2: Dependency accuracy of the McDonald model on 200 test sentences. When $\alpha = 0$, training only occurs on the supervised seed data. As α increases, we train based on confidence in our model's analysis of the unlabeled data. **Boldface** results are the best in their rows in a permutation test at the .05 level.

3.4 Comparison with EM

Perhaps the most popular statistical method for learning from incomplete data is the EM algorithm (Dempster et al., 1977). Since we cannot try EM on McDonald's conditional model, we ran some pilot experiments using the generative dependency model with valence (DMV) of Klein and Manning (2004). As in their experiments, and unlike the other experiments in the current paper, we restricted ourselves

Updated	M feat.	acc.	Updated	M feat.	acc.
all	15.5	64.3	none	0	60.9
seed	1.4	64.1	non-seed	14.1	44.7
non-lexical	3.5	64.4	lexical	12.0	59.9
non-bilex.	12.6	64.4	bilexical	2.9	61.0

Table 3: Using all features, dependency accuracy on German development data rose to 64.3% on bootstrapping. We show the contribution of different feature splits to the performance of this final model. For example, although this model was trained by updating all 15.5M feature weights, it performs as well if we *then* keep only the 1.4M features that appeared at least once in the seed set, zeroing out the weights of the others. We do as well as the full feature set if we keep only the 3.5M non-lexicalized features.

		% accuracy		
	train	Bulg.	German	Spanish
supervised	ML	74.2	80.0	71.3
	CL	77.5	79.3	75.0
semi-	EM	58.6	58.8	68.4
supervised	Conf.	80.0	80.5	76.7

Table 4: Dependency accuracy of the DMV model (Klein and Manning, 2004). Maximizing confidence using R_1 (Shannon) entropy improved performance over its own conditional likelihood (CL) baseline and over maximum likelihood (ML). EM degraded its ML baseline. Since these models were only trained and tested on sentences of 10 words or fewer, accuracy is much higher than the full results in Table 2.

to sentences of ten words or fewer and to part-of-speech sequences alone, without any lexical information. Since the DMV models projective trees, we ran experiments on three CoNLL corpora that had augmented their primary non-projective parses with alternate projective annotations: Bulgarian (Simov et al., 2005), German, and Spanish.

We performed supervised maximum likelihood and conditional likelihood estimation on a seed set of 100 sentences for each language. These models respectively initialized EM and confidence training on unlabeled data. We see (Table 4) that EM degrades the performance of its ML baseline. Merialdo (1994) saw a similar degradation over small (and large) seed sets in HMM POS tagging. We tried fixing and not fixing the feature expectations on the seed set during EM and show the former, better numbers. Confidence maximization improved over both its own conditional likelihood initializer and also over ML. We selected optimal smoothing parameters for all models and optimal α (equation (6)) and γ (equation (4)) for the confidence model on labeled held-out data.

4 Future Work

We hypothesize that qualitatively better bootstrapping results will require much larger unlabeled data sets. In scaling up bootstrapping to larger unlabeled training sets, we must carefully weight trade-offs between expanding coverage and introducing noise from out-of-domain data. We could also better exploit the data we have with richer models of syntax. In supervised dependency parsing, second-order edge features provide improvements (McDonald and Pereira, 2006; Riedel and Clarke, 2006); moreover, the feature-based approach is not limited to dependency parsing. Similar techniques could score parses in other formalisms, such as CFG or TAG. In this case, $\mathbf{f}$ extracts features from each of the derivation tree's rewrite rules (CFG) or elementary trees (TAG). In lexicalized formalisms, $\mathbf{f}$ will still be able to score lexical dependencies that are implicitly represented in the parse. Finally, we want to investigate whether larger training sets will provide traction for sparser cross-lingual and cross-domain features.

5 Conclusions

Feature-rich dependency models promise to help bootstrapping by providing many redundant features for the learner, and they can also cleanly incorporate cross-domain and cross-language information.

We explored bootstrapping feature-rich non-projective dependency parsers for Czech, German, and Spanish. Our bootstrapping method maximizes a linear combination of likelihood and confidence. In initial experiments on small datasets, this surpassed EM for training a simple feature-poor generative model, and also improved the performance of a feature-rich, conditionally estimated model where EM could not easily have been applied. For our models and training sets, more peaked measures of confidence, measured by Rényi entropy, outperformed smoother ones.

Acknowledgments

The authors thank the anonymous reviewers, Noah A. Smith, and Keith Hall for helpful comments, and Ryan McDonald for making his parsing code publicly available. This work was supported in part by NSF ITR grant IIS-0313193.

References

Steven Abney. 2004. Understanding the Yarowsky algorithm. *CL*, 30(3):365–395.

Cristoph Arndt. 2001. *Information Measures*. Springer.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*, pages 120–128.

A. Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In A. Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*, chapter 7. Kluwer.

Léon Bottou. 2003. Stochastic learning. In *Advanced Lectures in Machine Learning*, pages 146–168. Springer.

Matthew E. Brand. 1999. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11(5):1155–1182.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *TLT*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

M. Civit Torruella and M. A. Martí Antonín. 2002. Design principles for a Spanish treebank. In *TLT*.

Brooke Cowan, Ivona Kučerová, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *EMNLP*, pages 232–241.

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. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING*, pages 340–345.

H. N. Gabow, Z. Galil, T. H. Spencer, and R. E. Tarjan. 1986. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *ACL*, pages 961–968.

Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In *NIPS*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11:311–325.

Rebecca Hwa. 2000. Sample selection for statistical grammar induction. In *EMNLP*, pages 45–52.

Martin Jansche. 2005. Treebank transfer. In *IWPT*.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *COLING/ACL*, pages 209–216.

Damianos Karakos, Jason Eisner, Sanjeev Khudanpur, and Carey E. Priebe. 2007. Cross-instance tuning of unsupervised document clustering algorithms. In *HLT-NAACL*, pages 252–259.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*, pages 479–486.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *COLING-ACL*, pages 817–824.

T. Koo, A. Globerson, X. Carreras, and M. Collins. 2007. Structured prediction models via the Matrix-Tree Theorem. In *EMNLP-CoNLL*.

Gideon S. Mann and Andrew McCallum. 2007. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *HLT-NAACL*.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *EMNLP*, pages 44–52, July.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *ACL*, pages 337–344.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *IWPT*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *ACL*, pages 91–98.

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

Fuchun Peng, Ralph Weischedel, Ana Licuanan, and Jinxi Xu. 2005. Combining deep linguistics analysis and surface pattern learning: A hybrid approach to Chinese definitional question answering. In *HLT-EMNLP*, pages 307–314.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *EMNLP*, pages 129–137.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *CoNLL*.

Charles Schafer. 2006. *Translation Discovery Using Diverse Smilarity Measures*. Ph.D. thesis, Johns Hopkins University.

K. Simov, P. Osenova, A. Simov, and M. Kouylekov. 2005. Design and implementation of the Bulgarian HPSG-based treebank. In *Journal of Research on Language and Computation – Special Issue*. Kluwer.

David A. Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*, pages 23–30.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *EMNLP-CoNLL*.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *EACL*.

Ralph Weischedel. 2004. Extracting dynamic evidence networks. Technical Report AFRL-IF-RS-TR-2004-246, BBN Technologies, Cambridge, MA, December.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *ACL*, pages 207–216.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, pages 189–196.