Better Informed Training of Latent Syntactic Features

Markus Dreyer and **Jason Eisner**
Department of Computer Science / Center for Language and Speech Processing
Johns Hopkins University
3400 North Charles Street, Baltimore, MD 21218 USA
{markus,jason}@clsp.jhu.edu

Abstract

We study unsupervised methods for learning refinements of the nonterminals in a treebank. Following Matsuzaki et al. (2005) and Prescher (2005), we may for example split NP without supervision into NP[0] and NP[1], which behave differently. We first propose to learn a PCFG that adds such features to nonterminals in such a way that they respect patterns of linguistic feature passing: each node's nonterminal features are either identical to, or independent of, those of its parent. This linguistic constraint reduces runtime and the number of parameters to be learned. However, it did not yield improvements when training on the Penn Treebank. An orthogonal strategy was more successful: to improve the performance of the EM learner by treebank preprocessing and by annealing methods that split nonterminals selectively. Using these methods, we can maintain high parsing accuracy while dramatically reducing the model size.

1 Introduction

Treebanks never contain enough information; thus PCFGs estimated straightforwardly from the Penn Treebank (Bies et al., 1995) work only moderately well (Charniak, 1996). To address this problem, researchers have used heuristics to add more information. Eisner (1996), Charniak (1997), Collins (1997), and many subsequent researchers[1] annotated every node with lexical features passed up from its "head child," in order to more precisely reflect the node's "inside" contents. Charniak (1997) and Johnson (1998) annotated each node with its parent and grandparent nonterminals, to more precisely reflect its "outside" context. Collins (1996) split the sentence label S into two versions, representing sentences with and without subjects. He also modified the treebank to contain different labels for standard and for base noun phrases. Klein and Manning (2003) identified nonterminals that could valuably be split into fine-grained ones using hand-written linguistic rules. Their unlexicalized parser combined several such heuristics with rule markovization and reached a performance similar to early lexicalized parsers.

In all these cases, choosing which nonterminals to split, and how, was a matter of art. Ideally such splits would be learned automatically from the given treebank itself. This would be less costly and more portable to treebanks for new domains and languages. One might also hope that the automatically learned splits would be more effective.

Matsuzaki et al. (2005) introduced a model for such learning: PCFG-LA.[2] They used EM to induce fine-grained versions of a given treebank's nonterminals and rules. We present models that similarly learn to propagate fine-grained features through the tree, but only in certain linguistically motivated ways. Our models therefore allocate a supply of free parameters differently, allowing more fine-grained nonterminals but less fine-grained control over the probabilities of rewriting them. We also present simple methods for deciding selectively (during training) which nonterminals to split and how.

Section 2 describes previous work in finding hidden information in treebanks. Section 3 describes automatically induced feature grammars. We start by describing the PCFG-LA model, then introduce new models that use specific agreement patterns to propagate features through the tree. Section 4 describes annealing-like procedures for training latent-annotation models. Section 5 describes the motivation and results of our experiments. We finish by discussing future work and conclusions in sections 6–7.

[1] Not to mention earlier *non*-PCFG lexicalized statistical parsers, notably Magerman (1995) for the Penn Treebank.

[2] Probabilistic context-free grammar with latent annotations.

Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006), pages 317–326,
Sydney, July 2006.

Citation	Observed data	Hidden data
Collins (1997)	Treebank tree with head child annotated on each nonterminal	*No hidden data. Degenerate EM case.*
Lari and Young (1990)	Words	Parse tree
Pereira and Schabes (1992)	Words and partial brackets	Parse tree
Klein and Manning (2001)	Part-of-speech tags	Parse tree
Chiang and Bikel (2002)	Treebank tree	Head child on each nonterminal
Matsuzaki et al. (2005)	Treebank tree	Integer feature on each nonterminal
INHERIT model (this paper)	Treebank tree and head child heuristics	Integer feature on each nonterminal

Table 1: Observed and hidden data in PCFG grammar learning.

2 Partially supervised EM learning

The parameters of a PCFG can be learned with or without supervision. In the supervised case, the complete tree is observed, and the rewrite rule probabilities can be estimated directly from the observed rule counts. In the unsupervised case, only the words are observed, and the learning method must induce the whole structure above them. (See Table 1.)

In the *partially* supervised case we will consider, some part of the tree is observed, and the remaining information has to be induced. Pereira and Schabes (1992) estimate PCFG parameters from partially bracketed sentences, using the inside-outside algorithm to induce the missing brackets and the missing node labels. Some authors define a complete tree as one that specifies not only a label but also a "head child" for each node. Chiang and Bikel (2002) induces the missing head-child information; Prescher (2005) induces both the head-child information and the latent annotations we will now discuss.

3 Feature Grammars

3.1 The PCFG-LA Model

Staying in the partially supervised paradigm, the PCFG-LA model described in Matsuzaki et al. (2005) observe whole treebank trees, but learn an "annotation" on each nonterminal token—an unspecified and uninterpreted integer that distinguishes otherwise identical nonterminals. Just as Collins manually split the S nonterminal label into S and SG for sentences with and without subjects, Matsuzaki et al. (2005) split S into $\mathsf{S}[1]$, $\mathsf{S}[2]$, ..., $\mathsf{S}[L]$ where L is a predefined number—but they do it automatically and systematically, and not only for S but for every nonterminal. Their partially supervised learning procedure observes trees that are fully bracketed and fully labeled, except for the integer subscript used to annotate each node. After automatically inducing the annotations with EM, their resulting parser performs just as well as one learned from a treebank whose nonterminals were manually refined through linguistic and error analysis (Klein and Manning, 2003).

In Matsuzaki's PCFG-LA model, rewrite rules take the form

$$X[\alpha] \rightarrow Y[\beta]\ Z[\gamma] \tag{1}$$

in the binary case, and

$$X[\alpha] \rightarrow w \tag{2}$$

in the lexical case. The probability of a tree consisting of rules $r_1, r_2, \ldots$ is given by the probability of its root symbol times the conditional probabilities of the rules. The annotated tree T_1 in Fig. 1, for example, has the following probability:

$$\begin{aligned} P(T_1) \;=\; & P(\mathtt{ROOT} \rightarrow \mathsf{S}[2]) \\ & \times P(\mathsf{S}[2] \rightarrow \mathsf{NP}[1]\ \mathsf{VP}[3]) \\ & \times P(\mathsf{NP}[1] \rightarrow^* \mathtt{He}) \\ & \times P(\mathsf{VP}[3] \rightarrow^* \mathtt{loves\ cookies}) \end{aligned}$$

where, to simplify the notation, we use $P(X \rightarrow Y\ Z)$ to denote the conditional probability $P(Y\ Z \mid X)$ that a given node with label X will have children $Y\ Z$.

Degrees of freedom. We will want to compare models that have about the same size. Models with more free parameters have an inherent advantage on modeling copious data because of their greater

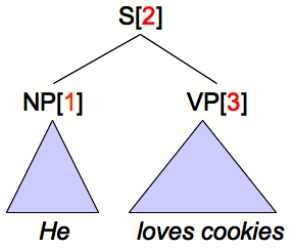

Figure 1: Treebank tree with annotations.

expressiveness. Models with fewer free parameters are easier to train accurately on sparse data, as well as being more efficient in space and often in time. Our question is therefore what can be accomplished with a given number of parameters.

How many free parameters in a PCFG-LA model? Such a model is created by annotating the nonterminals of a standard PCFG (extracted from the given treebank) with the various integers from 1 to L. If the original, "backbone" grammar has R_3 binary rules of the form $X \rightarrow Y\ Z$, then the resulting PCFG-LA model has $L^3 \times R_3$ such rules: $X[1] \rightarrow Y[1]\ Z[1]$, $X[1] \rightarrow Y[1]\ Z[2]$, $X[1] \rightarrow Y[2]\ Z[1]$, ..., $X[L] \rightarrow Y[L]\ Z[L]$. Similarly, if the backbone grammar has R_2 rules of the form $X \rightarrow Y$ the PCFG-LA model has $L^2 \times R_2$ such rules.[3] The number of R_1 terminal rules $X \rightarrow w$ is just multiplied by L.

The PCFG-LA has as many parameters to learn as rules: one probability per rule. However, not all these parameters are free, as there are $L \times N$ sum-to-one constraints, where N is the number of backbone nonterminals. Thus we have

$$L^3 R_3 + L^2 R_2 + L R_1 - L N \qquad (3)$$

degrees of freedom.

We note that Goodman (1997) mentioned possible ways to factor the probability 1, making independence assumptions in order to reduce the number of parameters.

Runtime. Assuming there are no unary rule cycles in the backbone grammar, bottom-up chart parsing of a length-n sentence at test time takes time proportional to $n^3 L^3 R_3 + n^2 L^2 R_2 + n L R_1$, by attempting to apply each rule everywhere in the sentence. (The dominating term comes from equation (4) of Table 2: we must loop over all n^3 triples i, j, k and all R_3 backbone rules $X \rightarrow Y Z$ and all L^3 triples α, β, γ.) As a function of n and L only, this is $O(n^3 L^3)$.

At training time, to induce the annotations on a given backbone tree with n nodes, one can run a constrained version of this algorithm that loops over only the n triples i, j, k that are consistent with the given tree (and considers only the single consistent backbone rule for each one). This takes time $O(n L^3)$, as does the inside-outside version we actually use to collect expected PCFG-LA rule counts for EM training.

We now introduce a model that is smaller, and has a lower runtime complexity, because it adheres to specified ways of propagating features through the tree.

3.2 Feature Passing: The INHERIT Model

Many linguistic theories assume that features get passed from the mother node to their children or some of their children. In many cases it is the head child that gets passed its feature value from its mother (e.g., Kaplan and Bresnan (1982), Pollard and Sag (1994)). In some cases the feature is passed to both the head and the non-head child, or perhaps even to the non-head alone.

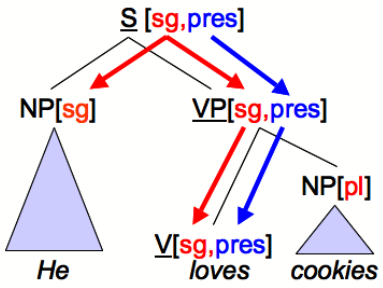

Figure 2: Features are passed to different children at different positions in the tree.

In the example in Fig. 2, the tense feature (*pres*) is always passed to the head child (underlined). How the number feature (*sg/pl*) is passed depends on the rewrite rule: S → NP VP passes it to both children, to enforce subject-verb agreement, while VP → V NP only passes it to the head child, since the object NP is free not to agree with the verb.

A feature grammar can incorporate such patterns of feature passing. We introduce additional parameters that define the probability of passing a feature to certain children. The head child of each node is given deterministically by the head rules of (Collins, 1996).

Under the INHERIT model that we propose, the

[3] We use unary rules of this form (e.g. the Treebank's S → NP) in our reimplementation of Matsuzaki's algorithm.

Model	Runtime and d.f.	Simplified equation for inside probabilities (ignores unary rules)
Matsuzaki et al. (2005)	test: $O(n^3L^3)$ train: $O(nL^3)$ d.f.: $L^3R_3 + L^2R_2 + LR_1 - LN$	$B_{X[\alpha]}(i,k) = \sum_{Y,\beta,Z,\gamma,j} P(X[\alpha] \rightarrow Y[\beta]\ Z[\gamma]) \times B_{Y[\beta]}(i,j) \times B_{Z[\gamma]}(j,k)$ (4)
INHERIT model (this paper)	test: $O(n^3L)$ train: $O(nL)$ d.f.: $L(R_3 + R_2 + R_1) + 3R_3 - N$	$B_{X[\alpha]}(i,k) = \sum_{Y,Z,j} P(X[\alpha] \rightarrow Y\ Z) \times \left(\begin{array}{l} P(\textbf{neither} \mid X,Y,Z) \times B_Y(i,j) \times B_Z(j,k)) \\ + P(\textbf{left} \mid X,Y,Z) \times B_{Y[\alpha]}(i,j) \times B_Z(j,k)) \\ + P(\textbf{right} \mid X,Y,Z) \times B_Y(i,j) \times B_{Z[\alpha]}(j,k)) \\ + P(\textbf{both} \mid X,Y,Z) \times B_{Y[\alpha]}Y(i,j) \times B_{Z[\alpha]}(j,k)) \end{array} \right)$ (5) $B_X(i,j) = \sum_\alpha P_{ann}(\alpha \mid X) \times B_{X[\alpha]}(i,j)$ (6) $P(\textbf{left} \mid X,Y,Z) = \begin{cases} P(\textbf{head} \mid X,Y,Z) & \text{if } Y \text{ heads } X \rightarrow \boldsymbol{Y}\ Z \\ P(\textbf{nonhead} \mid X,Y,Z) & \text{otherwise} \end{cases}$ (7) $P(\textbf{right} \mid X,Y,Z) = \begin{cases} P(\textbf{head} \mid X,Y,Z) & \text{if } Z \text{ heads } X \rightarrow Y\ \boldsymbol{Z} \\ P(\textbf{nonhead} \mid X,Y,Z) & \text{otherwise} \end{cases}$ (8)

Table 2: Comparison of the PCFG-LA model with the INHERIT model proposed in this paper. "d.f." stands for "degrees of freedom" (i.e., free parameters). The B terms are inside probabilities; to compute Viterbi parse probabilities instead, replace summation by maximization. Note the use of the intermediate quantity $B_X(i,j)$ to improve runtime complexity by moving some summations out of the inner loop; this is an instance of a "folding transformation" (Blatz and Eisner, 2006).

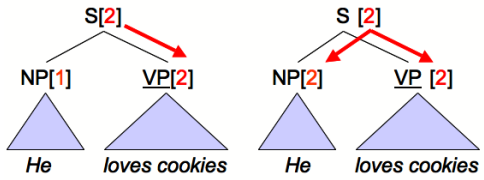

Figure 3: Two passpatterns. Left: T_2. The feature is passed to the head child (underlined). Right: T_3. The feature is passed to both children.

probabilities of tree T_2 in Fig. 3 are calculated as follows, with $P_{ann}(1 \mid NP)$ being the probability of annotating an NP with feature 1 if it does *not* inherit its parent's feature. The VP is boldfaced to indicate that it is the head child of this rule.

$$\begin{aligned} P(T_2) = {} & P(\texttt{ROOT} \rightarrow \texttt{S}[2]) \\ & \times P(\texttt{S}[2] \rightarrow \texttt{NP}\ \textbf{VP}) \\ & \times P(\textbf{pass to head} \mid \texttt{S} \rightarrow \texttt{NP}\ \textbf{VP}) \\ & \times P_{ann}(1 \mid \texttt{NP}) \times P(\texttt{NP}[1] \rightarrow^* \texttt{He}) \\ & \times P(\texttt{VP}[2] \rightarrow^* \texttt{loves cookies}) \end{aligned}$$

Tree T_3 in Fig. 3 has the following probability:

$$\begin{aligned} P(T_3) = {} & P(\texttt{ROOT} \rightarrow \texttt{S}[2]) \\ & \times P(\texttt{S}[2] \rightarrow \texttt{NP}\ \textbf{VP}) \\ & \times P(\textbf{pass to both} \mid \texttt{S} \rightarrow \texttt{NP}\ \textbf{VP}) \\ & \times P(\texttt{NP}[2] \rightarrow^* \texttt{He}) \\ & \times P(\texttt{VP}[2] \rightarrow^* \texttt{loves cookies}) \end{aligned}$$

In T_2, the subject NP chose feature 1 or 2 independent of its parent S, according to the distribution $P_{ann}(\cdot \mid \texttt{NP})$. In T_3, it was constrained to inherit its parent's feature 2.

Degrees of freedom. The INHERIT model may be regarded as containing all the same rules (see (1)) as the PCFG-LA model. However, these rules' probabilities are now collectively determined by a smaller set of shared parameters.[4] That is because the distribution of the child features β and γ no longer depends arbitrarily on the rest of the rule. β is either equal to α, or chosen independently of everything but Y.

The model needs probabilities for $L \times R_3$ binary-rule parameters like $P(\texttt{S}[2] \rightarrow \texttt{NP}\ \textbf{VP})$ above, as well as $L \times R_2$ unary-rule and $L \times R_1$ lexical-rule parameters. None of these consider the annotations on the children. They are subject to $L \times N$ sum-to-one constraints.

The model also needs $4 \times R_3$ passpattern probabilities like $P(\textbf{pass to head} \mid X \rightarrow YZ)$ above, with R_3 sum-to-one constraints, and $L \times N$ non-inherited annotation parameters $P_{ann}(\alpha|X)$, with N sum-to-one constraints.

Adding these up and canceling the two $L \times N$

[4] The reader may find it useful to write out the probability $P(X[\alpha] \rightarrow Y[\beta]\ Z[\gamma])$ in terms of the parameters described below. Like equation (5), it is $P(X[\alpha] \rightarrow Y\ Z)$ times a sum of up to 4 products, corresponding to the 4 passpattern cases.

terms, the INHERIT model has

$$L(R_3 + R_2 + R_1) + 3R_3 - N \qquad (9)$$

degrees of freedom. Thus for a typical grammar where R_3 dominates, we have reduced the number of free parameters from about $L^3 R_3$ to only about LR_3.

Runtime. We may likewise reduce an L^3 factor to L in the runtime. Table 2 shows dynamic programming equations for the INHERIT model. By exercising care, they are able to avoid summing over all possible values of β and γ within the inner loop. This is possible because when they are not inherited, they do not depend on X, Y, Z, or α.

3.3 Multiple Features

The INHERIT model described above is linguistically naive in several ways. One problem (see section 6 for others) is that each nonterminal has only a single feature to pass. Linguists, however, usually annotate each phrase with multiple features. Our example tree in Fig. 2 was annotated with both tense and number features, with different inheritance patterns.

As a step up from INHERIT, we propose an INHERIT2 model where each nonterminal carries two features. Thus, we will have $L^6 R_3$ binary rules instead of $L^3 R_3$. However, we assume that the two features choose their passpatterns independently, and that when a feature is not inherited, it is chosen independently of the other feature. This keeps the number of parameters down. In effect, we are defining

$$\begin{aligned} & P(X[\alpha][\rho] \rightarrow Y[\beta][\sigma]\ Z[\gamma][\tau]) \\ = \ & P(X[\alpha][\rho] \rightarrow Y\ Z) \\ & \times P_1(\beta, \gamma \mid X[\alpha] \rightarrow Y\ Z) \\ & \times P_2(\sigma, \tau \mid X[\rho] \rightarrow Y\ Z) \end{aligned}$$

where P_1 and P_2 choose child features as if they were separate single-feature INHERIT models.

We omit discussion of dynamic programming speedups for INHERIT2. Empirically, the hope is that the two features when learned with the EM algorithm will pick out different linguistic properties of the constituents in the treebank tree.

4 Annealing-Like Training Approaches

Training latent PCFG models, like training most other unsupervised models, requires non-convex optimization. To find good parameter values, it is often helpful to train a simpler model first and use its parameters to derive a starting guess for the harder optimization problem. A well-known example is the training of the IBM models for statistical machine translation (Berger et al., 1994).

In this vein, we did an experiment in which we gradually increased L during EM training of the PCFG-LA and INHERIT models. Whenever the training likelihood began to converge, we *manually* and *globally* increased L, simply doubling or tripling it (see "clone all" in Table 3 and Fig. 5). The probability of $X[\alpha] \rightarrow Y[\beta]Z[\gamma]$ under the new model was initialized to be proportional to the probability of $X[\alpha \bmod L] \rightarrow Y[\beta \bmod L]Z[\gamma \bmod L]$ (where L refers to the old L),[5] times a random "jitter" to break symmetry.

In a second annealing experiment ("clone some") we addressed a weakness of the PCFG-LA and INHERIT models: They give every nonterminal the same number of latent annotations. It would seem that different coarse-grained nonterminals in the original Penn Treebank have different degrees of impurity (Klein and Manning, 2003). There are linguistically many kinds of `NP`, which are differentially selected for by various contexts and hence are worth distinguishing. By contrast, `-LRB-` is almost always realized as a left parenthesis and may not need further refinement. Our "clone some" annealing starts by training a model with L=2 to convergence. Then, instead of cloning all nonterminals as in the previous annealing experiments, we clone only those that have seemed to benefit most from their previous refinement. This benefit is measured by the Jensen-Shannon divergence of the two distributions $P(X[0] \rightarrow \cdots)$ and $P(X[1] \rightarrow \cdots)$. The

[5] Notice that as well as cloning $X[\alpha]$, this procedure multiplies by 4, 2, and 1 the number of binary, unary, and lexical rules that rewrite $X[\alpha]$. To leave the backbone grammar unchanged, we should have scaled down the probabilities of such rules by 1/4, 1/2, and 1 respectively. Instead, we simply scaled them all down by the same proportion. While this temporarily changes the balance of probability among the three kinds of rules, EM immediately corrects this balance on the next training iteration to match the observed balance on the treebank trees—hence the one-iteration downtick in Figure 5).

Jensen-Shannon divergence is defined as

$$D(q,r) = \frac{1}{2}\left(D\left(q \,||\, \frac{q+r}{2}\right) + D\left(r \,||\, \frac{q+r}{2}\right)\right)$$

These experiments are a kind of "poor man's version" of the deterministic annealing clustering algorithm (Pereira et al., 1993; Rose, 1998), which gradually increases the number of clusters during the clustering process. In deterministic annealing, one starts in principle with a very large number of clusters, but maximizes likelihood only under a constraint that the joint distribution $p(point, cluster)$ must have very high entropy. This drives all of the cluster centroids to coincide exactly, redundantly representing just one effective cluster. As the entropy is permitted to decrease, some of the cluster centroids find it worthwhile to drift apart.[6] In future work, we would like to apply this technique to split nonterminals gradually, by initially requiring high-entropy parse forests on the training data and slowly relaxing this constraint.

5 Experiments

5.1 Setup

We ran several experiments to compare the INHERIT with the PCFG-LA model and look into the effect of different Treebank preprocessing and the annealing-like procedures.

We used sections 2–20 of the Penn Treebank 2 Wall Street Journal corpus (Marcus et al., 1993) for training, section 22 as development set and section 23 for testing. Following Matsuzaki et al. (2005), words occurring fewer than 4 times in the training corpus were replaced by unknown-word symbols that encoded certain suffix and capitalization information.

All experiments used simple add-lambda smoothing (λ=0.1) during the reestimation step (M step) of training.

Binarization and Markovization. Before extracting the backbone PCFG and running the constrained inside-outside (EM) training algorithm, we preprocessed the Treebank using center-parent binarization Matsuzaki et al. (2005). Besides making the rules at most binary, this preprocessing also helpfully enriched the backbone nonterminals. For all but the first ("Basic") experiments, we also enriched the nonterminals with order-1 horizontal and order-2 vertical markovization (Klein and Manning, 2003).[7] Figure 4 shows what a multiple-child structure $X \rightarrow A\ B\ \boldsymbol{H}\ C\ D$ looks like after binarization and markovization. The binarization process starts at the head of the sentence and moves to the right, inserting an auxiliary node for each picked up child, then moving to the left. Each auxiliary node consists of the parent label, the direction (L or R) and the label of the child just picked up.

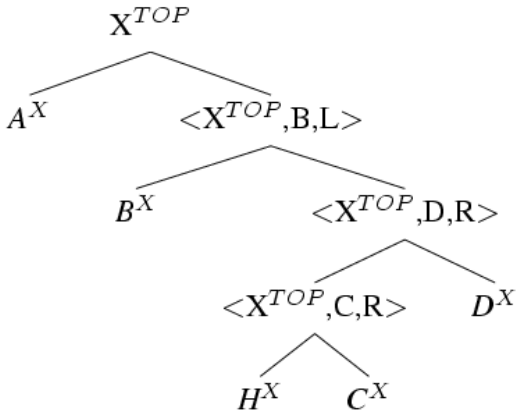

Figure 4: Horizontal and vertical markovization and center-parent binarization of the rule $X \rightarrow A\ B\ \boldsymbol{H}\ C\ D$ where H is the head child.

Initialization. The backbone PCFG grammar was read off the altered Treebank, and the initial annotated grammar was created by creating several versions of every rewrite rule. The probabilities of these newly created rules are uniform and proportional to the original rule, multiplied by a random epsilon factor uniformly sampled from [.9999,1.0001] to break symmetry.

5.2 Decoding

To test the PCFG learned by a given method, we attempted to recover the *unannotated* parse of each sentence in the development set. We then scored these parses by debinarizing or demarkovizing them, then measuring their precision and recall of the labeled constituents from the gold-standard Treebank parses.

[6] In practice, each very large group of centroids (effective cluster) is represented by just two, until such time as those two drift apart to represent separate effective clusters—then each is cloned.

[7] The vertical markovization was applied *before* binarization. – Matsuzaki et al. (2005) used a markovized grammar to get a better unannotated parse forest during decoding, but they did not markovize the training data.

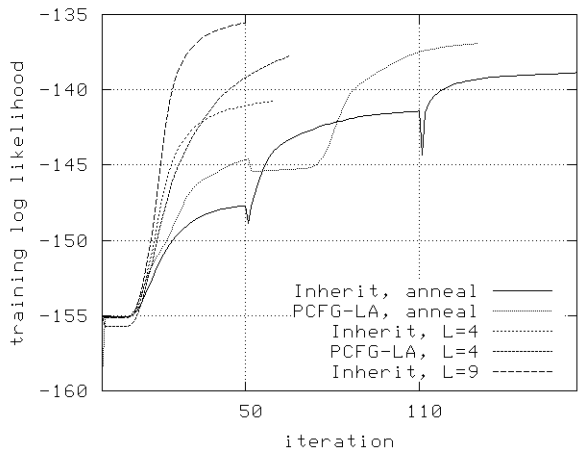

Figure 5: Log_e-likelihood during training. The two "anneal" curves use the "clone all" method. We increased L after iteration 50 and, for the INHERIT model, iteration 110. The downward spikes in the two annealed cases are due to perturbation of the model parameters (footnote 5).

An unannotated parse's probability is the total probability, under our learned PCFG, of all of its annotated refinements. This total can be efficiently computed by the constrained version of the inside algorithm in Table 2.

How do we obtain the unannotated parse whose *total* probability is greatest? It does not suffice to find the single best annotated parse and then strip off the annotations. Matsuzaki et al. (2005) note that the best annotated parse is in fact NP-hard to find. We use their reranking approximation. A 1000-best list for each sentence in the decoding set was created by parsing with our markovized *un*annotated grammar and extracting the 1000 best parses using the k-best algorithm 3 described in Huang and Chiang (2005). Then we chose the most probable of these 1000 unannotated parses under our PCFG, first finding the total probability of each by using the the constrained inside algorithm as explained above.[8]

5.3 Results and Discussion

Table 3 summarizes the results on development and test data. [9] Figure 5 shows the training log-likelihoods.

First, markovization of the Treebank leads to striking improvements. The "Basic" block of experiments in Table 3 used non-markovized grammars, as in Matsuzaki et al. (2005). The next block of experiments, introducing markovized grammars, shows a considerable improvement. This is not simply because markovization increases the number of parameters: markovization with $L = 2$ already beats basic models that have much higher L and far more parameters.

Evidently, markovization pre-splits the labels in the trees in a reasonable way, so EM has less work to do. This is not to say that markovization eliminates the need for hidden annotations: with markovization, going from L=1 to L=2 increases the parsing accuracy even more than without it.

Second, our "clone all" training technique (shown in the next block of Table 3) did not help performance and may even have hurt slightly. Here we initialized the L=2x2 model with the trained L=2 model for PCFG-LA, and the L=3x3 model with the L=3 and the L=3x3x3 model with the L=3x3 model.

Third, our "clone some" training technique appeared to work. On PCFG-LA, the L<2x2 condition (i.e., train with L=2 and then clone some) matched the performance of L=4 with 30% fewer parameters. On INHERIT, L<2x2 beat L=4 with 8% fewer parameters. In these experiments, we used the average divergence as a threshold: X[0] and X[1] are split again if the divergence of their rewrite distributions is higher than average.

Fourth, our INHERIT model was a disappointment. It generally performed slightly worse than PCFG-LA when given about as many degrees of freedom. This was also the case on some cursory experiments on smaller training corpora. It is tempting to conclude that INHERIT simply adopted overly strong linguistic constraints, but relaxing those constraints by moving to the INHERIT2 model did not seem to help. In our one experiment with INHERIT2 (not shown in Table 3), using 2 features that can each take L=2 values (d.f.: 212,707) obtains an F_1 score of only 83.67—worse than 1 feature taking L=4 values.

5.4 Analysis: What was learned by INHERIT?

INHERIT did seem to discover "linguistic" features, as intended, even though this did not improve parse accuracy. We trained INHERIT and PCFG-LA models (both L=2, non-markovized) and noticed the following.

[8] For the first set of experiments, in which the models were trained on a simple non-markovized grammar, the 1000-best trees had to be "demarkovized" before our PCFG was able to rescore them.

[9] All results are reported on sentences of 40 words or less.

	PCFG-LA					INHERIT				
	L	d.f.	LP	LR	F_1	L	d.f.	LP	LR	F_1
Basic	1	24,226	76.99	74.51	75.73	1	35,956	76.99	74.51	75.73
	2	72,392	81.22	80.67	80.94	2	60,902	79.42	77.58	78.49
	4	334,384	83.53	83.39	83.46	12	303,162	82.41	81.55	81.98
	8	2,177,888	85.43	85.05	85.24	80	1,959,053	83.99	83.02	83.50
Markov.	1	41,027	79.95	78.43	79.18	1	88,385	79.95	78.43	79.18
						2	132,371	83.85	82.23	83.03
	2	178,264	85.70	84.37	85.03	3	176,357	85.04	83.60	84.31
						4	220,343	85.30	84.06	84.68
	3	506,427	86.44	85.19	85.81	9	440,273	86.16	85.12	85.64
	4	1,120,232	87.09	85.71	86.39	26	1,188,035	86.55	85.55	86.05
Clone all	2	178,264	85.70	84.37	85.03	3	176,357	85.04	83.60	84.31
						3x3	440,273	85.99	84.88	85.43
	2x2	1,120,232	87.06	85.49	86.27	3x3x3	1,232,021	86.65	85.70	86.17
Cl.some	2	178,264	85.70	84.37	85.03	2	132,371	83.85	82.23	83.03
	<2x2	789,279	**87.17**	**85.71**	**86.43**	<2x2	203,673	85.49	84.45	84.97
						<2x2x2	314,999	85.57	84.60	85.08

Table 3: Results on the development set: labeled precision (LP), labeled recall (LR), and their harmonic mean (F_1). "Basic" models are trained on a non-markovized treebank (as in Matsuzaki et al. (2005)); all others are trained on a markovized treebank. The best model (PCFG-LA with "clone some" annealing, F_1=86.43) has also been decoded on the final test set, reaching P/R=86.94/85.40 (F_1=86.17).

We used both models to assign the most-probable annotations to the gold parses of the development set. Under the INHERIT model, NP$[0]$ vs. NP$[1]$ constituents were 21% plural vs. 41% plural. Under PCFG-LA this effect was weaker (30% vs. 39%), although it was significant in both (Fisher's exact test, $p < 0.001$). Strikingly, under the INHERIT model, the NP's were 10 times more likely to pass this feature to both children (Fisher's, $p < 0.001$)—just as we would expect for a number feature, since the determiner and head noun of an NP must agree.

The INHERIT model also learned to use feature value 1 for "tensed auxiliary." The VP$[1]$ nonterminal was far more likely than VP$[0]$ to expand as V VP, where V represents any of the tensed verb preterminals VBZ, VBG, VBN, VBD, VBP. Furthermore, these expansion rules had a very strong preference for "pass to head," so that the left child would also be annotated as a tensed auxiliary, typically causing it to expand as a form of be, have, or do. In short, the feature ensured that it was genuine auxiliary verbs that subcategorized for VP$'s$.

(The PCFG-LA model actually arranged the same behavior, e.g. similarly preferring VBZ$[1]$ in the auxiliary expansion rule VP $\rightarrow$ VBZ VP. The difference is that the PCFG-LA model was able to express this preference directly without propagating the $[1]$ up to the VP parent. Hence neither VP$[0]$ nor VP$[1]$ became strongly associated with the auxiliary rule.)

Many things are equally learned by both models: They learn the difference between subordinating conjunctions (*while, if*) and prepositions (*under, after*), putting them in distinct groups of the original IN tag, which typically combine with sentences and noun phrases, respectively. Both models also split the conjunction CC into two distinct groups: a group of conjunctions starting with an upper-case letter at the beginning of the sentence and a group containing all other conjunctions.

6 Future Work: Log-Linear Modeling

Our approach in the INHERIT model made certain strict independence assumptions, with no backoff. The choice of a particular passpattern, for example, depends on all and only the three nonterminals X, Y, Z. However, given sparse training data, sometimes it is advantageous to back off to smaller amounts of contextual information; the nonterminal X or Y might alone be sufficient to predict the passpattern.

A very reasonable framework for handling this issue is to model $P(X[\alpha] \rightarrow Y[\beta]\ Z[\gamma])$ with a log-linear model.[10] Feature functions would consider the values of variously sized, overlapping subsets of $X, Y, Z, \alpha, \beta, \gamma$. For example, a certain feature might fire when $X[\alpha] =$ NP$[1]$ **and** $Z[\gamma] =$ N$[2]$. This approach can be extended to the multi-feature case, as in INHERIT2.

Inheritance as in the INHERIT model can then be expressed by features like $\alpha = \beta$, or $\alpha = \beta$ **and** $X =$ VP. During early iterations, we could use a prior to encourage a strong positive weight on these inheritance features, and gradually relax this bias—akin to the "structural annealing" of (Smith and Eisner, 2006).

When modeling the lexical rule $P(X[\alpha] \rightarrow w)$, we could use features that consider the spelling of the word w in conjunction with the value of α. Thus, we might learn that $V[1]$ is particularly likely to rewrite as a word ending in -s. Spelling features that are predictable from string context are important clues to the existence and behavior of the hidden annotations we wish to induce.

A final remark is that "inheritance" does not necessarily have to mean that $\alpha = \beta$. It is enough that α and β should have high mutual information, so that one can be predicted from the other; they do not actually have to be represented by the same integer. More broadly, we might like α to have high mutual information with the pair (β, γ). One might try using this sort of intuition directly in an unsupervised learning procedure (Elidan and Friedman, 2003).

7 Conclusions

We have discussed "informed" techniques for inducing latent syntactic features. Our INHERIT model tries to constrain the way in which features are passed through the tree. The motivation for this approach is twofold: First, we wanted to capture the linguistic insight that features follow certain patterns in propagating through the tree. Second, we wanted to make it statistically feasible and computationally tractable to increase L to higher values than in the PCFG-LA model. The hope was that the learning process could then make finer distinctions and learn more fine-grained information. However, it turned out that the higher values of L did not compensate for the perhaps overly constrained model. The results on English parsing rather suggest that it is the similarity in degrees of freedom (e.g., INHERIT with L=3x3x3 and PCFG-LA with L=2x2) that produces comparable results.

Substantial gains were achieved by using markovization and splitting only selected nonterminals. With these techniques we reach a parsing accuracy similar to Matsuzaki et al. (2005), but with an order of magnitude less parameters, resulting in more efficient parsing. We hope to get more wins in future by using more sophisticated annealing techniques and log-linear modeling techniques.

[10] This affects EM training only by requiring a convex optimization at the M step (Riezler, 1998).

Acknowledgments

This paper is based upon work supported by the National Science Foundation under Grant No. 0313193. We are grateful to Takuya Matsuzaki for providing details about his implementation of PCFG-LA, and to Noah Smith and the anonymous reviewers for helpful comments.

References

A. Berger, P. Brown, S. Pietra, V. Pietra, J. Lafferty, H. Printz, and L. Ures. 1994. The CANDIDE system for machine translation.

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing guidelines for Treebank II style: Penn Treebank project. Technical Report MS-CIS-95-06, University of Pennsylvania, January.

John Blatz and Jason Eisner. 2006. Transforming parsing algorithms and other weighted logic programs. In *Proceedings of the 11th Conference on Formal Grammar*.

Eugene Charniak. 1996. Tree-bank grammars. In *Proceedings of the 13th National Conference on Artificial Intelligence*.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 598–603.

David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *COLING 2002: The 17th International Conference on Computational Linguistics*, Taipei.

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *ACL-96*, pages 184–191, Santa Cruz, CA. ACL.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Madrid. Association for Computational Linguistics.

Jason Eisner, Eric Goldlust, and Noah A. Smith. 2005. Compiling comp ling: Weighted dynamic programming and the Dyna language. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 281–290, Vancouver, October.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING96: Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345, Copenhagen. Center for Sprogteknologi.

Gal Elidan and Nir Friedman. 2003. The information bottleneck EM algorithm. In *Proceedings of UAI*.

Joshua Goodman. 1997. Probabilistic feature grammars. In *Proceedings of the 5th International Workshop on Parsing Technologies*, pages 89–100, MIT, Cambridge, MA, September.

L. Huang and D. Chiang. 2005. Parsing and k-best algorithms. In *Proc. of IWPT*.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA.

Dan Klein and Christopher D. Manning. 2001. Distributional phrase structure induction. In *The Fifth Conference on Natural Language Learning*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In Erhard Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

K. Lari and S. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

David M. Magerman. 1995. Statistical Decision-Tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, Mass.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330, June.

Takuya Matsuzaki, Yusuke Miyao, and Junichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, University of Michigan.

Fernando Pereira and Yves Schabes. 1992. Inside-Outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Meeting of the Association for Computational Linguistics*, pages 128–135, Newark. University of Delaware.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *Proceedings of ACL*, Ohio State University.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

Detlef Prescher. 2005. Head-driven PCFGs with latent-head statistics. In *Proceedings of the 9th International Workshop on Parsing Technologies*, pages 115–124, Vancouver, BC, Canada, October.

Stefan Riezler. 1998. Statistical inference and probabilistic modeling for constraint-based NLP. In B. Schröder, W. Lenders, W. Hess, and T. Portele, editors, *Computers, Linguistics, and Phonetics between Language and Speech: Proceedings of the 4th Conference on Natural Language Processing (KONVENS'98)*, pages 111–124, Bonn. Lang.

Kenneth Rose. 1998. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 80:2210–2239, November.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of ACL*.