Probabilistic CFG with latent annotations

Takuya Matsuzaki† Yusuke Miyao† Jun'ichi Tsujii†‡
†Graduate School of Information Science and Technology, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033
‡CREST, JST(Japan Science and Technology Agency)
Honcho 4-1-8, Kawaguchi-shi, Saitama 332-0012
{matuzaki, yusuke, tsujii}@is.s.u-tokyo.ac.jp

Abstract

This paper defines a generative probabilistic model of parse trees, which we call PCFG-LA. This model is an extension of PCFG in which non-terminal symbols are augmented with latent variables. Fine-grained CFG rules are automatically induced from a parsed corpus by training a PCFG-LA model using an EM-algorithm. Because exact parsing with a PCFG-LA is NP-hard, several approximations are described and empirically compared. In experiments using the Penn WSJ corpus, our automatically trained model gave a performance of 86.6% (F_1, sentences $\leq$ 40 words), which is comparable to that of an unlexicalized PCFG parser created using extensive manual feature selection.

1 Introduction

Variants of PCFGs form the basis of several broad-coverage and high-precision parsers (Collins, 1999; Charniak, 1999; Klein and Manning, 2003). In those parsers, the strong conditional independence assumption made in vanilla treebank PCFGs is weakened by annotating non-terminal symbols with many 'features' (Goodman, 1997; Johnson, 1998). Examples of such features are head words of constituents, labels of ancestor and sibling nodes, and subcategorization frames of lexical heads. Effective features and their good combinations are normally explored using trial-and-error.

This paper defines a generative model of parse trees that we call PCFG with latent annotations (PCFG-LA). This model is an extension of PCFG models in which non-terminal symbols are annotated with latent variables. The latent variables work just like the features attached to non-terminal symbols. A fine-grained PCFG is automatically induced from parsed corpora by training a PCFG-LA model using an EM-algorithm, which replaces the manual feature selection used in previous research.

The main focus of this paper is to examine the effectiveness of the automatically trained models in parsing. Because exact inference with a PCFG-LA, i.e., selection of the most probable parse, is NP-hard, we are forced to use some approximation of it. We empirically compared three different approximation methods. One of the three methods gives a performance of 86.6% (F_1, sentences $\leq$ 40 words) on the standard test set of the Penn WSJ corpus.

Utsuro et al. (1996) proposed a method that automatically selects a proper level of generalization of non-terminal symbols of a PCFG, but they did not report the results of parsing with the obtained PCFG. Henderson's parsing model (Henderson, 2003) has a similar motivation as ours in that a derivation history of a parse tree is compactly represented by induced hidden variables (hidden layer activation of a neural network), although the details of his approach is quite different from ours.

2 Probabilistic model

PCFG-LA is a generative probabilistic model of parse trees. In this model, an observed parse tree is considered as an incomplete data, and the corre-

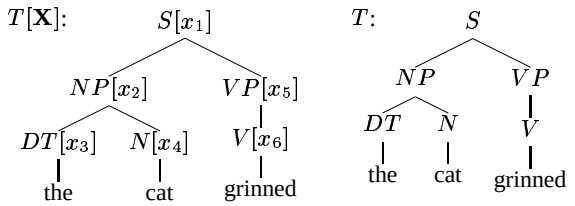


Figure 1: Tree with latent annotations $T[\mathbf{X}]$ (complete data) and observed tree T (incomplete data).

sponding complete data is a tree with *latent annotations*. Each non-terminal node in the complete data is labeled with a complete symbol of the form $A[x]$, where A is the non-terminal symbol of the corresponding node in the observed tree and x is a *latent annotation symbol*, which is an element of a fixed set H.

A complete/incomplete tree pair of the sentence, "*the cat grinned*," is shown in Figure 2. The complete parse tree, $T[\mathbf{X}]$ (left), is generated through a process just like the one in ordinary PCFGs, but the non-terminal symbols in the CFG rules are annotated with latent symbols, $\mathbf{X} = (x_1, x_2, \dots)$. Thus, the probability of the complete tree ($T[\mathbf{X}]$) is

$$\begin{aligned} &P(T[\mathbf{X}]) \\ &= \pi(S[x_1]) \times \beta(S[x_1] \rightarrow NP[x_2]VP[x_5]) \\ &\times \beta(NP[x_2] \rightarrow DT[x_3]N[x_4]) \\ &\times \beta(DT[x_3] \rightarrow the) \times \beta(N[x_4] \rightarrow cat) \\ &\times \beta(VP[x_5] \rightarrow V[x_6]) \times \beta(V[x_6] \rightarrow grinned), \end{aligned}$$

where $\pi(S[x_1])$ denotes the probability of an occurrence of the symbol $S[x_1]$ at a root node and $\beta(r)$ denotes the probability of a CFG rule r. The probability of the observed tree $P(T)$ is obtained by summing $P(T[\mathbf{X}])$ for all the assignments to latent annotation symbols, $\mathbf{X}$:

$$P(T) = \sum_{x_1 \in H} \sum_{x_2 \in H} \cdots \sum_{x_6 \in H} P(T[\mathbf{X}]). \quad (1)$$

Using dynamic programming, the theoretical bound of the time complexity of the summation in Eq. 1 is reduced to be proportional to the number of non-terminal nodes in a parse tree. However, the calculation at node n still has a cost that exponentially grows with the number of n's daughters because we must sum up the probabilities of $|H|^{d+1}$ combinations of latent annotation symbols for a node with d daughters. We thus took a kind of transformation/detransformation approach, in which a tree is binarized before parameter estimation and restored to its original form after parsing. The details of the binarization are explained in Section 4.

Using syntactically annotated corpora as training data, we can estimate the parameters of a PCFG-LA model using an EM algorithm. The algorithm is a special variant of the inside-outside algorithm of Pereira and Schabes (1992). Several recent work also use similar estimation algorithm as ours, i.e, inside-outside re-estimation on parse trees (Chiang and Bikel, 2002; Shen, 2004).

The rest of this section precisely defines PCFG-LA models and briefly explains the estimation algorithm. The derivation of the estimation algorithm is largely omitted; see Pereira and Schabes (1992) for details.

2.1 Model definition

We define a PCFG-LA $\mathcal{M}$ as a tuple $\mathcal{M} = \langle N_{\mathrm{nt}}, N_{\mathrm{t}}, H, R, \pi, \beta \rangle$, where

N_{nt} : a set of observable non-terminal symbols
N_{t} : a set of terminal symbols
H : a set of latent annotation symbols
R: a set of observable CFG rules
$\pi(A[x])$: the probability of the occurrence of a complete symbol $A[x]$ at a root node
$\beta(r)$: the probability of a rule $r \in R[H]$.

We use $A, B, \dots$ for non-terminal symbols in N_{nt}; $w_1, w_2, \dots$ for terminal symbols in N_{t}; and $x, y, \dots$ for latent annotation symbols in H. $N_{\mathrm{nt}}[H]$ denotes the set of complete non-terminal symbols, i.e., $N_{\mathrm{nt}}[H] = \{A[x] \mid A \in N_{\mathrm{nt}}, x \in H\}$. Note that latent annotation symbols are not attached to terminal symbols.

In the above definition, R is a set of CFG rules of observable (i.e., not annotated) symbols. For simplicity of discussion, we assume that R is a CNF grammar, but extending to the general case is straightforward. $R[H]$ is the set of CFG rules of complete symbols, such as $V[x] \rightarrow$ *grinned* or $S[x] \rightarrow NP[y]VP[z]$. More precisely,

$$\begin{aligned} R[H] = &\{(A[x] \rightarrow w) \mid (A \rightarrow w) \in R; x \in H\} \cup \\ &\{(A[x] \rightarrow B[y]C[z]) \mid (A \rightarrow BC) \in R; x, y, z \in H\}. \end{aligned}$$

We assume that non-terminal nodes in a parse tree T are indexed by integers $i = 1, \ldots, m$, starting from the root node. A complete tree is denoted by $T[\mathbf{X}]$, where $\mathbf{X} = (x_1, \ldots, x_m) \in H^m$ is a vector of latent annotation symbols and x_i is the latent annotation symbol attached to the i-th non-terminal node.

We do not assume any structured parametrizations in β and π; that is, each $\beta(r)$ $(r \in R[H])$ and $\pi(A[x])$ $(A[x] \in N_{\mathrm{nt}}[H])$ is itself a parameter to be tuned. Therefore, an annotation symbol, say, x, generally does not express any commonalities among the complete non-terminals annotated by x, such as $A[x], B[x], etc.$

The probability of a complete parse tree $T[\mathbf{X}]$ is defined as

$$P(T[\mathbf{X}]) = \pi(A_1[x_1]) \prod_{r \in D_{T[\mathbf{X}]}} \beta(r), \qquad (2)$$

where $A_1[x_1]$ is the label of the root node of $T[\mathbf{X}]$ and $D_{T[\mathbf{X}]}$ denotes the multiset of annotated CFG rules used in the generation of $T[\mathbf{X}]$. We have the probability of an observable tree T by marginalizing out the latent annotation symbols in $T[\mathbf{X}]$:

$$P(T) = \sum_{X \in H^m} \pi(A_1[x_1]) \prod_{r \in D_{T[\mathbf{X}]}} \beta(r), \qquad (3)$$

where m is the number of non-terminal nodes in T.

2.2 Forward-backward probability

The sum in Eq. 3 can be calculated using a dynamic programming algorithm analogous to the forward algorithm for HMMs. For a sentence $w_1 w_2 \ldots w_n$ and its parse tree T, backward probabilities $b_T^i(x)$ are recursively computed for the i-th non-terminal node and for each $x \in H$. In the definition below, $N_i \in N_{\mathrm{nt}}$ denotes the non-terminal label of the i-th node.

- If node i is a pre-terminal node above a terminal symbol w_j, then $b_T^i(x) = \beta(N_i[x] \to w_j)$.
- Otherwise, let j and k be the two daughter nodes of i. Then

$$b_T^i(x) = \sum_{x_j, x_k \in H} \begin{array}{c} \beta(N_i[x] \to N_j[x_j] N_k[x_k]) \\ \times\ b_T^j(x_j)\ b_T^k(x_k). \end{array}$$

Using backward probabilities, $P(T)$ is calculated as $P(T) = \sum_{x_1 \in H} \pi(N_1[x_1])\ b_T^1(x_1)$.

We define forward probabilities $f_T^i(x)$, which are used in the estimation described below, as follows:

- If node i is the root node (i.e., i = 1), then $f_T^i(x) = \pi(N_i[x])$.
- If node i has a right sibling k, let j be the mother node of i. Then

$$f_T^i(x) = \sum_{x_j, x_k \in H} \begin{array}{c} \beta(N_j[x_j] \to N_i[x] N_k[x_k]) \\ \times\ f_T^j(x_j)\ b_T^k(x_k). \end{array}$$

- If node i has a left sibling, $f_T^i(x)$ is defined analogously.

2.3 Estimation

We now derive the EM algorithm for PCFG-LA, which estimates the parameters $\theta = (\beta, \pi)$. Let $\mathbf{T} = \{T_1, T_2, \ldots\}$ be the training set of parse trees and $N_1^i, \ldots, N_{m_i}^i$ be the labels of non-terminal nodes in T_i. Like the derivations of the EM algorithms for other latent variable models, the update formulas for the parameters, which update the parameters from θ to $\theta' = (\beta', \pi')$, are obtained by constrained optimization of $Q(\theta'|\theta)$, which is defined as

$$Q(\theta'|\theta) = \sum_{T_i \in \mathbf{T}} \sum_{\mathbf{X}_i \in H^{m_i}} P_\theta(\mathbf{X}_i|T_i) \log P_{\theta'}(T_i[\mathbf{X}_i]),$$

where P_θ and $P_{\theta'}$ denote probabilities under θ and θ', and $P(\mathbf{X}|T)$ is the conditional probability of latent annotation symbols given an observed tree T, i.e., $P(\mathbf{X}|T) = P(T[\mathbf{X}])/P(T)$. Using the Lagrange multiplier method and re-arranging the results using the backward and forward probabilities, we obtain the update formulas in Figure 2.

3 Parsing with PCFG-LA

In theory, we can use PCFG-LAs to parse a given sentence w by selecting the most probable parse:

$$T_{best} = \operatorname*{argmax}_{T \in G(w)} P(T|w) = \operatorname*{argmax}_{T \in G(w)} P(T), \qquad (4)$$

where $G(w)$ denotes the set of possible parses for w under the observable grammar R. While the optimization problem in Eq. 4 can be efficiently solved

$$\beta'(A[x] \to B[y]C[z]) = Z_{A[x]}^{-1} \sum_{T_i \in \mathbf{T}} P(T_i)^{-1} \times \sum_{(j,k,l) \in \text{Covered}(T_i, A \to BC)} f_{T_i}^j(x)\ \beta(A[x] \to B[y]C[z])\ b_{T_i}^k(y)\ b_{T_i}^l(z)$$

$$\beta'(A[x] \to w) = Z_{A[x]}^{-1} \sum_{T_i \in \mathbf{T}} P(T_i)^{-1} \sum_{j \in \text{Covered}(T_i, A \to w)} f_{T_i}^j(x)\ \beta(A[x] \to w)$$

$$\pi'(A[x]) = |\mathbf{T}|^{-1} \sum_{T_i \in \text{Root}(\mathbf{T}, A)} P(T_i)^{-1} \pi(A[x])\ b_{T_i}^1(x)$$

$$Z_{A[x]} = \sum_{T_i \in \mathbf{T}} P(T_i)^{-1} \sum_{j \in \text{Labeled}(T_i, A)} f_{T_i}^j(x)\ b_{T_i}^j(x)$$

$$\text{Covered}(T_i, A \to BC) = \{(j,k,l) \mid N_j^i \to N_k^i N_l^i \in D_{T_i}; (N_j^i, N_k^i, N_l^i) = (A,B,C)\}$$

$$\text{Covered}(T_i, A \to w) = \{j \mid N_j^i \to w \in D_{T_i}; N_j^i = A\}$$

$$\text{Labeled}(T_i, A) = \{j \mid N_j^i = A\}$$

$$\text{Root}(\mathbf{T}, A) = \{T_i \in \mathbf{T} \mid \text{the root of } T_i \text{ is labeled with } A\}$$

Figure 2: Parameter update formulas.

for PCFGs using dynamic programming algorithms, the sum-of-products form of $P(T)$ in PCFG-LA models (see Eq. 2 and Eq. 3) makes it difficult to apply such techniques to solve Eq. 4.

Actually, the optimization problem in Eq. 4 is NP-hard for general PCFG-LA models. Although we omit the details, we can prove the NP-hardness by observing that a stochastic tree substitution grammar (STSG) can be represented by a PCFG-LA model in a similar way to one described by Goodman (1996a), and then using the NP-hardness of STSG parsing (Simaán, 2002).

The difficulty of the exact optimization in Eq. 4 forces us to use some approximations of it. The rest of this section describes three different approximations, which are empirically compared in the next section. The first method simply limits the number of candidate parse trees compared in Eq. 4; we first create N-best parses using a PCFG and then, within the N-best parses, select the one with the highest probability in terms of the PCFG-LA. The other two methods are a little more complicated, and we explain them in separate subsections.

3.1 Approximation by Viterbi complete trees

The second approximation method selects the best *complete* tree $T'[\mathbf{X}']$, that is,

$$T'[\mathbf{X}'] = \operatorname*{argmax}_{T \in G(w), \mathbf{X} \in H^{|\mathbf{X}|}} P(T[\mathbf{X}]). \qquad (5)$$

We call $T'[\mathbf{X}']$ a Viterbi complete tree. Such a tree can be obtained in $O(|w|^3)$ time by regarding the PCFG-LA as a PCFG with annotated symbols.[1]

The observable part of the Viterbi complete tree $T'[\mathbf{X}']$ (i.e., T') does not necessarily coincide with the best observable tree T_{best} in Eq. 4. However, if T_{best} has some 'dominant' assignment $\mathbf{Y}$ to its latent annotation symbols such that $P(T_{best}[\mathbf{Y}]) \approx P(T_{best})$, then $P(T') \approx P(T_{best})$ because $P(T_{best}[\mathbf{Y}]) \leq P(T'[\mathbf{X}'])$ and $P(T'[\mathbf{X}']) \leq P(T')$, and thus T' and T_{best} are almost equally 'good' in terms of their marginal probabilities.

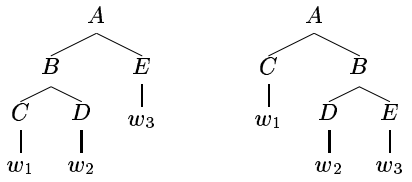
3.2 Viterbi parse in approximate distribution

In the third method, we approximate the true distribution $P(T|w)$ by a cruder distribution $Q(T|w)$, and then find the tree with the highest $Q(T|w)$ in polynomial time. We first create a packed representation of $G(w)$ for a given sentence w.[2] Then, the approximate distribution $Q(T|w)$ is created using the packed forest, and the parameters in $Q(T|w)$ are adjusted so that $Q(T|w)$ approximates $P(T|w)$ as closely as possible. The form of $Q(T|w)$ is that of a product of the parameters, just like the form of a PCFG model, and it enables us to use a Viterbi algorithm to select the tree with the highest $Q(T|w)$.

A packed forest is defined as a tuple $\langle I, \delta \rangle$. The first component, I, is a multiset of chart items of the form (A, b, e). A chart item $(A, b, e) \in I$ indicates that there exists a parse tree in $G(w)$ that contains a constituent with the non-terminal label A that spans

[1] For efficiency, we did not actually parse sentences with $R[H]$ but selected a Viterbi complete tree from a packed representation of candidate parses in the experiments in Section 4.

[2] In practice, fully constructing a packed representation of $G(w)$ has an unrealistically high cost for most input sentences. Alternatively, we can use a packed representation of a subset of $G(w)$, which can be obtained by parsing with beam thresholding, for instance. An approximate distribution $Q(T|w)$ on such subsets can be derived in almost the same way as one for the full $G(w)$, but the conditional distribution, $P(T|w)$, is renormalized so that the total mass for the subset sums to 1.



$$I = \{i_1, i_2, i_3, i_4, i_5, i_6\}$$
$$i_1 = (A, 1, 3), i_2 = (B, 1, 2), i_3 = (B, 2, 3),$$
$$i_4 = (C, 1, 1), i_5 = (D, 2, 2), i_6 = (E, 3, 3)$$
$$\delta(i_1) = \{(i_2, i_6), (i_3, i_4)\}$$
$$\delta(i_2) = \{(i_4, i_5)\},\ \ \delta(i_3) = \{(i_5, i_6)\}$$
$$\delta(i_4) = \{w_1\}, \delta(i_5) = \{w_2\}, \delta(i_6) = \{w_3\}$$

Figure 3: Two parse trees and packed representation of them.

from the b-th to e-th word in w. The second component, δ, is a function on I that represents dominance relations among the chart items in I; $\delta(i)$ is a set of possible daughters of i if i is not a pre-terminal node, and $\delta(i) = \{w_k\}$ if i is a pre-terminal node above w_k. Two parse trees for a sentence $w = w_1 w_2 w_3$ and a packed representation of them are shown in Figure 3.

We require that each tree $T \in G(w)$ has a unique representation as a set of connected chart items in I. A packed representation satisfying the uniqueness condition is created using the CKY algorithm with the observable grammar R, for instance.

The approximate distribution, $Q(T|w)$, is defined as a PCFG, whose CFG rules R_w is defined as $R_w = \{(i \to \eta) \mid i \in I; \eta \in \delta(i)\}$. We use $q(r)$ to denote the rule probability of rule $r \in R_w$ and $q_r(i)$ to denote the probability with which $i \in I$ is generated as a root node. We define $Q(T|w)$ as

$$Q(T|w) = q_r(i_1) \prod_{k=1}^{m} q(i_k \to \eta_k),$$

where the set of connected items $\{i_1, \ldots, i_m\} \subset I$ is the unique representation of T.

To measure the closeness of approximation by $Q(T|w)$, we use the 'inclusive' KL-divergence, $KL(P||Q)$ (Frey et al., 2000):

$$KL(P||Q) = \sum_{T \in G(w)} P(T|w) \log \frac{P(T|w)}{Q(T|w)}.$$

Minimizing $KL(P||Q)$ under the normalization constraints on q_r and q yields closed form solutions for q_r and q, as shown in Figure 4.

P_{in} and P_{out} in Figure 4 are similar to ordinary inside/outside probabilities. We define P_{in} as follows:

- If $i = (A, k, k) \in I$ is a pre-terminal node above w_k, then $P_{\text{in}}(i[x]) = \beta(A[x] \to w_k)$.
- Otherwise,

$$P_{\text{in}}(i[x]) = \sum_{jk \in \delta(i)} \sum_{y,z \in H} \begin{array}{l} \beta(A[x] \to B_j[y] C_k[z]) \\ \times P_{\text{in}}(j[y]) P_{\text{in}}(k[z]), \end{array}$$

  where B_j and C_k denote non-terminal symbols of chart items j and k.

The outside probability, P_{out}, is calculated using P_{in} and PCFG-LA parameters along the packed structure, like the outside probabilities for PCFGs.

Once we have computed $q(i \to \eta)$ and $q_r(i)$, the parse tree T that maximizes $Q(T|w)$ is found using a Viterbi algorithm, as in PCFG parsing.

Several parsing algorithms that also use inside-outside calculation on packed chart have been proposed (Goodman, 1996b; Simaán, 2003; Clark and Curran, 2004). Those algorithms optimize some evaluation metric of parse trees other than the posterior probability $P(T|w)$, e.g., (expected) labeled constituent recall or (expected) recall rate of dependency relations contained in a parse. It is in contrast with our approach where (approximated) posterior probability is optimized.

4 Experiments

We conducted four sets of experiments. In the first set of experiments, the degree of dependency of trained models on initialization was examined because EM-style algorithms yield different results with different initial values of parameters. In the second set of experiments, we examined the relationship between model types and their parsing performances. In the third set of experiments, we compared the three parsing methods described in the previous section. Finally, we show the result of a parsing experiment using the standard test set.

We used sections 2 through 20 of the Penn WSJ corpus as training data and section 21 as heldout data. The heldout data was used for early stopping; i.e., the estimation was stopped when the rate

- If $i_1 \in I$ is not a pre-terminal node, for each $\eta = i_2 i_3 \in \delta(i_1)$, let A, B, and C be non-terminal symbols of i_1, i_2, and i_3. Then,
$$q(i_1 \to \eta) = \frac{\sum_{x \in H} \sum_{y \in H} \sum_{z \in H} P_{\text{out}}(i_1[x]) \beta(A[x] \to B[y]C[z]) P_{\text{in}}(i_2[y]) P_{\text{in}}(i_3[z])}{\sum_{x \in H} P_{\text{out}}(i_1[x]) P_{\text{in}}(i_1[x])}.$$
- If $i \in I$ is a pre-terminal node above word w_k, then $q(i \to w_k) = 1$.
- If $i \in I$ is a root node, let A be the non-terminal symbol of i. Then $q_r(i) = \frac{1}{P(w)} \sum_{x \in H} \pi(A[x]) P_{\text{in}}(i[x])$.

Figure 4: Optimal parameters of approximate distribution Q.

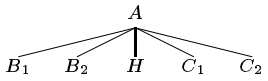


Figure 5: Original subtree.

of increase in the likelihood of the heldout data became lower than a certain threshold. Section 22 was used as test data in all parsing experiments except in the final one, in which section 23 was used. We stripped off all function tags and eliminated empty nodes in the training and heldout data, but any other pre-processing, such as comma raising or base-NP marking (Collins, 1999), was not done except for binarizations.

4.1 Dependency on initial values

To see the degree of dependency of trained models on initializations, four instances of the same model were trained with different initial values of parameters.[3] The model used in this experiment was created by CENTER-PARENT binarization and $|H|$ was set to 16. Table 1 lists training/heldout data log-likelihood per sentence (LL) for the four instances and their parsing performances on the test set (section 22). The parsing performances were obtained using the approximate distribution method in Section 3.2. Different initial values were shown to affect the results of training to some extent (Table 1).

[3]The initial value for an annotated rule probability, $\beta(A[x] \to B[y]C[z])$, was created by randomly multiplying the maximum likelihood estimation of the corresponding PCFG rule probability, $P(A \to BC)$, as follows:

$$\beta(A[x] \to B[y]C[z]) = Z_A^{-1} e^{\gamma} P(A \to BC),$$

where γ is a random number that is uniformly distributed in $[-\log 3, \log 3]$ and Z_A is a normalization constant.

	1	2	3	4	average $\pm \sigma$
training LL	-115	-114	-115	-114	-114 $\pm$ 0.41
heldout LL	-114	-115	-115	-114	-114 $\pm$ 0.29
LR	86.7	86.3	86.3	87.0	86.6 $\pm$ 0.27
LP	86.2	85.6	85.5	86.6	86.0 $\pm$ 0.48

Table 1: Dependency on initial values.

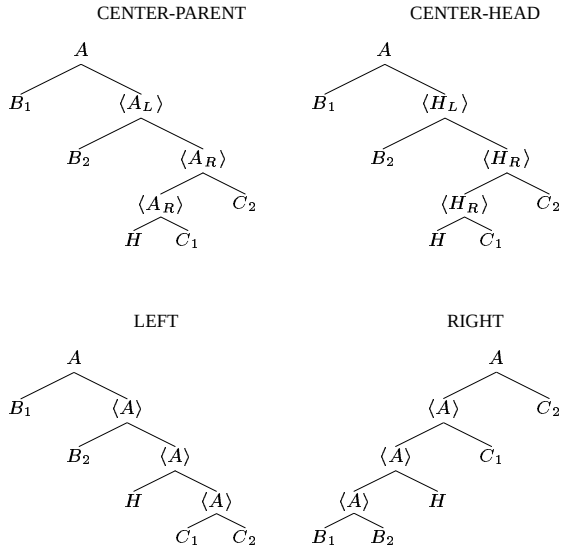


Figure 6: Four types of binarization (H: head daughter).

4.2 Model types and parsing performance

We compared four types of binarization. The original form is depicted in Figure 5 and the results are shown in Figure 6. In the first two methods, called CENTER-PARENT and CENTER-HEAD, the head-finding rules of Collins (1999) were used. We obtained an observable grammar R for each model by reading off grammar rules from the binarized training trees. For each binarization method, PCFG-LA models with different numbers of latent annotation symbols, $|H| = 1, 2, 4, 8$, and 16, were trained.

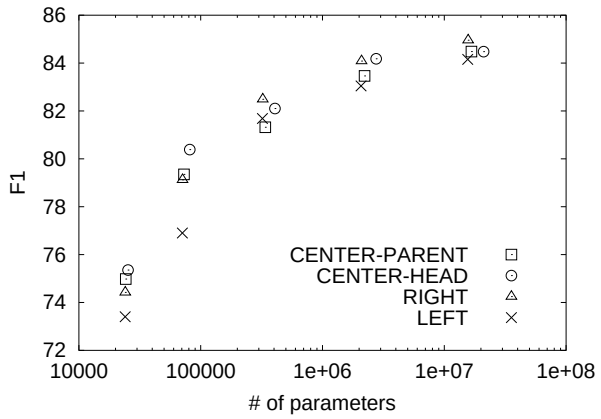


Figure 7: Model size vs. parsing performance.

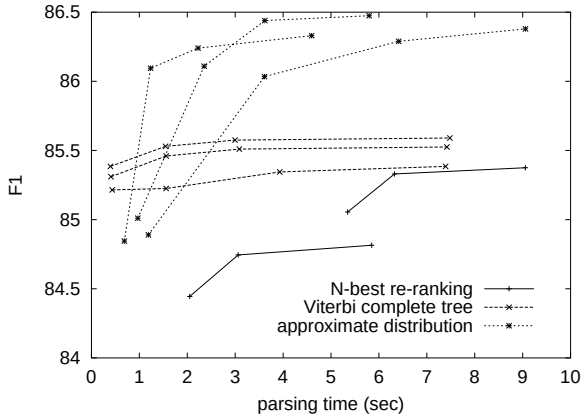


Figure 8: Comparison of parsing methods.

The relationships between the number of parameters in the models and their parsing performances are shown in Figure 7. Note that models created using different binarization methods have different numbers of parameters for the same $|H|$. The parsing performances were measured using F_1 scores of the parse trees that were obtained by re-ranking of 1000-best parses by a PCFG.

We can see that the parsing performance gets better as the model size increases. We can also see that models of roughly the same size yield similar performances regardless of the binarization scheme used for them, except the models created using LEFT binarization with small numbers of parameters ($|H| =$ 1 and 2). Taking into account the dependency on initial values at the level shown in the previous experiment, we cannot say that any single model is superior to the other models when the sizes of the models are large enough.

The results shown in Figure 7 suggest that we could further improve parsing performance by increasing the model size. However, both the memory size and the training time are more than linear in $|H|$, and the training time for the largest ($|H| = 16$) models was about 15 hours for the models created using CENTER-PARENT, CENTER-HEAD, and LEFT and about 20 hours for the model created using RIGHT. To deal with larger (e.g., $|H|$ = 32 or 64) models, we therefore need to use a model search that reduces the number of parameters while maintaining the model's performance, and an approximation during training to reduce the training time.

4.3 Comparison of parsing methods

The relationships between the average parse time and parsing performance using the three parsing methods described in Section 3 are shown in Figure 8. A model created using CENTER-PARENT with $|H| = 16$ was used throughout this experiment.

The data points were made by varying configurable parameters of each method, which control the number of candidate parses. To create the candidate parses, we first parsed input sentences using a PCFG[4], using beam thresholding with beam width α. The data points on a line in the figure were created by varying α with other parameters fixed. The first method re-ranked the N-best parses enumerated from the chart after the PCFG parsing. The two lines for the first method in the figure correspond to N = 100 and N = 300. In the second and the third methods, we removed all the dominance relations among chart items that did not contribute to any parses whose PCFG-scores were higher than $\gamma P_{\max}$, where $P_{\max}$ is the PCFG-score of the best parse in the chart. The parses remaining in the chart were the candidate parses for the second and the third methods. The different lines for the second and the third methods correspond to different values of γ.

The third method outperforms the other two methods unless the parse time is very limited (i.e., $<$ 1

[4]The PCFG used in creating the candidate parses is roughly the same as the one that Klein and Manning (2003) call a 'markovised PCFG with vertical order = 2 and horizontal order = 1' and was extracted from Section 02-20. The PCFG itself gave a performance of 79.6/78.5 LP/LR on the development set. This PCFG was also used in the experiment in Section 4.4.

$\leq$ 40 words	LR	LP	CB	0 CB
This paper	86.7	86.6	1.19	61.1
Klein and Manning (2003)	85.7	86.9	1.10	60.3
Collins (1999)	88.5	88.7	0.92	66.7
Charniak (1999)	90.1	90.1	0.74	70.1
$\leq$ 100 words	LR	LP	CB	0 CB
This paper	86.0	86.1	1.39	58.3
Klein and Manning (2003)	85.1	86.3	1.31	57.2
Collins (1999)	88.1	88.3	1.06	64.0
Charniak (1999)	89.6	89.5	0.88	67.6

Table 2: Comparison with other parsers.

sec is required), as shown in the figure. The superiority of the third method over the first method seems to stem from the difference in the number of candidate parses from which the outputs are selected.[5] The superiority of the third method over the second method is a natural consequence of the consistent use of $P(T)$ both in the estimation (as the objective function) and in the parsing (as the score of a parse).

4.4 Comparison with related work

Parsing performance on section 23 of the WSJ corpus using a PCFG-LA model is shown in Table 2. We used the instance of the four compared in the second experiment that gave the best results on the development set. Several previously reported results on the same test set are also listed in Table 2.

Our result is lower than the state-of-the-art lexicalized PCFG parsers (Collins, 1999; Charniak, 1999), but comparable to the unlexicalized PCFG parser of Klein and Manning (2003). Klein and Manning's PCFG is annotated by many linguistically motivated features that they found using extensive manual feature selection. In contrast, our method induces all parameters automatically, except that manually written head-rules are used in binarization. Thus, our method can extract a considerable amount of hidden regularity from parsed corpora. However, our result is worse than the lexicalized parsers despite the fact that our model has access to words in the sentences. It suggests that certain types of information used in those lexicalized parsers are hard to be learned by our approach.

[5] Actually, the number of parses contained in the packed forest is more than 1 million for over half of the test sentences when $\alpha = 10^{-4}$ and $\gamma = 10^{-3}$, while the number of parses for which the first method can compute the exact probability in a comparable time (around 4 sec) is only about 300.

References

Eugene Charniak. 1999. A maximum-entropy-inspired parser. Technical Report CS-99-12.

David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *Proc. COLING*, pages 183–189.

Stephen Clark and James R. Curran. 2004. Parsing the wsj using ccg and log-linear models. In *Proc. ACL*, pages 104–111.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Brendan J. Frey, Relu Patrascu, Tommi Jaakkola, and Jodi Moran. 2000. Sequentially fitting "inclusive" trees for inference in noisy-OR networks. In *Proc. NIPS*, pages 493–499.

Joshua Goodman. 1996a. Efficient algorithms for parsing the DOP model. In *Proc. EMNLP*, pages 143–152.

Joshua Goodman. 1996b. Parsing algorithms and metric. In *Proc. ACL*, pages 177–183.

Joshua Goodman. 1997. Probabilistic feature grammars. In *Proc. IWPT*.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proc. HLT-NAACL*, pages 103–110.

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

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. ACL*, pages 423–430.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proc. ACL*, pages 128–135.

Libin Shen. 2004. Nondeterministic LTAG derivation tree extraction. In *Proc. TAG+7*, pages 199–203.

Khalil Simaán. 2002. Computational complexity of probabilistic disambiguation. *Grammars*, 5(2):125–151.

Khalil Simaán. 2003. On maximizing metrics for syntactic disambiguation. In *Proc. IWPT*.

Takehito Utsuro, Syuuji Kodama, and Yuji Matsumoto. 1996. Generalization/specialization of context free grammars based-on entropy of non-terminals. In *Proc. JSAI (in Japanese)*, pages 327–330.