Head-Driven PCFGs with Latent-Head Statistics

Detlef Prescher
Institute for Logic, Language and Computation
University of Amsterdam
prescher@science.uva.nl

Abstract

Although *state-of-the-art* parsers for natural language are lexicalized, it was recently shown that an *accurate* unlexicalized parser for the Penn tree-bank can be simply read off a manually refined tree-bank. While *lexicalized* parsers often suffer from sparse data, *manual mark-up* is costly and largely based on individual linguistic intuition. Thus, across domains, languages, and tree-bank annotations, a fundamental question arises: Is it possible to *automatically* induce an *accurate* parser from a tree-bank without resorting to full lexicalization? In this paper, we show how to induce head-driven probabilistic parsers with latent heads from a tree-bank. Our automatically trained parser has a performance of 85.7% (LP/LR F_1), which is already better than that of early *lexicalized* ones.

1 Introduction

State-of-the-art statistical parsers for natural language are based on probabilistic grammars acquired from transformed tree-banks. The method of transforming the tree-bank is of major influence on the accuracy and coverage of the statistical parser. The most important tree-bank transformation in the literature is lexicalization: Each node in a tree is labeled with its head word, the most important word of the constituent under the node (Magerman (1995), Collins (1996), Charniak (1997), Collins (1997), Carroll and Rooth (1998), etc.). It turns out, however, that lexicalization is not unproblematic: First, there is evidence that full lexicalization does not carry over across different tree-banks for other languages, annotations or domains (Dubey and Keller, 2003). Second, full lexicalization leads to a serious sparse-data problem, which can only be solved by sophisticated smoothing and pruning techniques.

Recently, Klein and Manning (2003) showed that a carefully performed linguistic mark-up of the tree-bank leads to almost the same performance results as lexicalization. This result is attractive since unlexicalized grammars are easy to estimate, easy to parse with, and time- and space-efficient: Klein and Manning (2003) do not smooth grammar-rule probabilities, except unknown-word probabilities, and they do not prune since they are able to determine the most probable parse of each *full* parse forest. Both facts are noteworthy in the context of statistical parsing with a tree-bank grammar. A drawback of their method is, however, that manual linguistic mark-up is not based on abstract rules but rather on individual linguistic intuition, which makes it difficult to repeat their experiment and to generalize their findings to languages other than English.

Is it possible to automatically acquire a more refined probabilistic grammar from a given tree-bank without resorting to full lexicalization? We present a novel method that is able to induce a parser that is located between two extremes: a fully-lexicalized parser on one side *versus* an accurate unlexicalized parser based on a manually refined tree-bank on the other side.

In short, our method is based on the same linguistic principles of headedness as other methods: We do believe that lexical information represents an important knowledge source. To circumvent data sparseness resulting from full lexicalization with words, we simply follow the suggestion of various advanced linguistic theories, e.g. Lexical-Functional Grammar (Bresnan and Kaplan, 1982), where more complex categories based on feature combinations represent the lexical effect. We complement this by a learning paradigm: lexical entries carry latent information to be used as head information, and this head information is induced from the tree-bank.

In this paper, we study two different latent-head models, as well as two different estimation methods: The first model is built around completely hidden heads, whereas the second one uses relatively fine-grained combinations of Part-Of-Speech (POS) tags with hidden extra-information; The first estimation method selects a head-driven probabilistic context-free grammar (PCFG) by exploiting latent-head distributions for each node in the tree-bank, whereas the second one is more traditional, reading off the grammar from the tree-bank annotated with the most probable latent heads only. In other words, both models and estimation methods differ in the degree of information incorporated into them as prior knowledge. In general, it can be expected that the better (sharper or richer, or more accurate) the information is, the better the induced grammar will be. Our empirical results, however, are surprising: First, estimation with latent-head distributions outperforms estimation with most-probable-head annotation. Second, modeling with completely hidden heads is almost as good as modeling with latent heads based on POS tags, and moreover, results in much smaller grammars.

We emphasize that our task is to automatically induce a more refined grammar based on a few linguistic principles. With automatic refinement it is harder to guarantee improved performance than with manual refinements (Klein and Manning, 2003) or with refinements based on direct lexicalization (Magerman (1995), Collins (1996), Charniak (1997), etc.). If, however, our refinement provides improved performance then it has a clear advantage: it is automatically induced, which suggests that it is applicable across different domains, languages and tree-bank annotations.

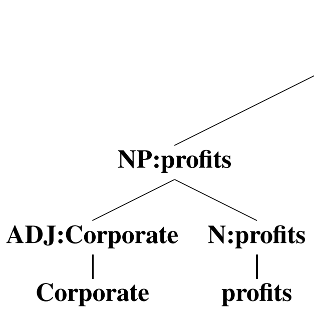
Applying our method to the benchmark Penn tree-bank Wall-Street Journal, we obtain a refined probabilistic grammar that significantly improves over the original tree-bank grammar and that shows performance that is on par with early work on lexicalized probabilistic grammars. This is a promising result given the hard task of automatic induction of improved probabilistic grammars.

2 Head Lexicalization

As previously shown (Charniak (1997), Collins (1997), Carroll and Rooth (1998), etc.), Context-Free Grammars (CFGs) can be transformed to lexicalized CFGs, provided that a head-marking scheme for rules is given. The basic idea is that the head marking on rules is used to project lexical items up a chain of nodes. Figure 1 displays an example.

In this Section, we focus on the approaches of Charniak (1997) and Carroll and Rooth (1998). These approaches are especially attractive for us for two reasons: First, both approaches make use of an *explicit linguistic grammar*. By contrast, alternative approaches, like Collins (1997), apply an additional transformation to each tree in the tree-bank, splitting each rule into small parts, which finally results in a new grammar covering many more sentences than the explicit one. Second, Charniak (1997) and Carroll and Rooth (1998) rely on almost the same lexicalization technique. In fact, the significant difference between them is that, in one case, a lexicalized version of the *tree-bank grammar* is learned from a corpus of trees (supervised learning), whereas, in the other case, a lexicalized version of a *manually written CFG* is learned from a a text corpus (unsupervised learning). As we will see in Section 3, our approach is a blend of these approaches in that it aims at unsupervised learning of a (latent-head-) lexicalized version of the tree-bank grammar.

Starting with Charniak (1997), Figure 2 displays an internal rule as it is used in the parse in Figure1, and its probability as defined by Charniak. Here, H is the head-child of the rule, which inherits the head h from its parent C. The children D_1:d_1, ..., D_m:d_m and D_{m+1}:d_{m+1}, ..., D_{m+n}:d_{m+n} are left and right modifiers of H. Either n or m may be zero, and $n = m = 0$ for unary rules. Because the probabilities occurring in Charniak's definition are already so specific that there is no real chance of obtaining the data empirically, they are smoothed by deleted interpolation:



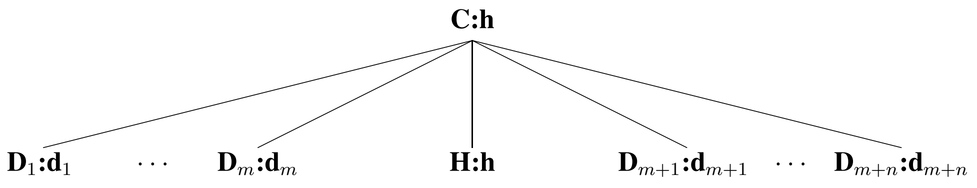
Internal Rules:

S:rose	$\longrightarrow$	NP:profits VP:rose PUNC:.
NP:profits	$\longrightarrow$	ADJ:Corporate N:profits
VP:rose	$\longrightarrow$	V:rose

Lexical Rules:

ADJ:Corporate	$\longrightarrow$	Corporate
N:profits	$\longrightarrow$	profits
V:rose	$\longrightarrow$	rose
PUNC:.	$\longrightarrow$	.

Figure 1: Parse tree, and a list of the rules it contains (Charniak, 1997)


C:h
$\mathbf{D}_1$:$\mathbf{d}_1$
. . .
$\mathbf{D}_m$:$\mathbf{d}_m$
H:h
$\mathbf{D}_{m+1}$:$\mathbf{d}_{m+1}$
. . .
$\mathbf{D}_{m+n}$:$\mathbf{d}_{m+n}$

$$p_{\text{CHARNIAK97}}(\text{ **this local tree** }) \;=\; p(\; r \mid \mathbf{C}, h, \mathbf{C}_p \;) \;\times\; \prod_{i=1}^{n+m} p(\; d_i \mid \mathbf{D}_i, \mathbf{C}, h \;)$$

(r **is the *unlexicalized* rule,**
$\mathbf{C}_p$ is C's parent category)

Figure 2: Internal rule, and its probability (Charniak, 1997)

$$\begin{aligned} p(\, r \mid \text{C},\, h,\, \text{C}_p \,) \;=\;& \lambda_1 \cdot \hat{p}(\, r \mid \text{C},\, h,\, \text{C}_p \,) \\ &+ \lambda_2 \cdot \hat{p}(\, r \mid \text{C},\, h \,) \\ &+ \lambda_3 \cdot \hat{p}(\, r \mid \text{C},\, \text{class}(h) \,) \\ &+ \lambda_4 \cdot \hat{p}(\, r \mid \text{C},\, \text{C}_p \,) \\ &+ \lambda_5 \cdot \hat{p}(\, r \mid \text{C} \,) \end{aligned}$$

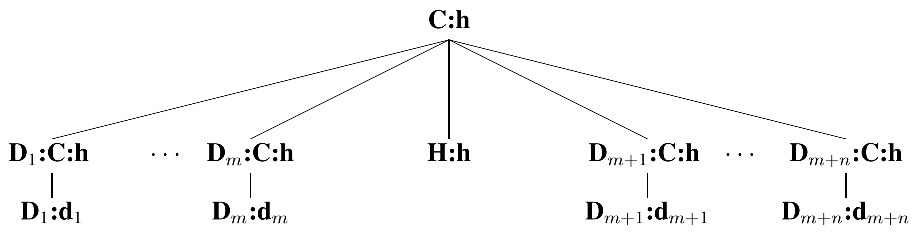
$$\begin{aligned} p(\, d \mid \text{D},\, \text{C},\, h \,) \;=\;& \lambda_1 \cdot \hat{p}(\, d \mid \text{D},\, \text{C},\, h \,) \\ &+ \lambda_2 \cdot \hat{p}(\, d \mid \text{D},\, \text{C},\, \text{class}(h) \,) \\ &+ \lambda_3 \cdot \hat{p}(\, d \mid \text{D},\, \text{C} \,) \\ &+ \lambda_4 \cdot \hat{p}(\, d \mid \text{D} \,) \end{aligned}$$

Here, class(h) denotes a class for the head word h. Charniak takes these word classes from an *external* distributional clustering model, but does not describe this model in detail.

An at a first glance different lexicalization technique is described in Carroll and Rooth (1998). In their approach, a grammar transformation is used to lexicalize a manually written grammar. The key step for understanding their model is to imagine that the rule in Figure 2 is transformed to a *sub-tree*, the one displayed in Figure 3. After this transformation, the sub-tree probability is simply calculated with the PCFG's standard model; The result is also displayed in the figure. Comparing this probability with the probability that Charniak assigns to the rule itself, we see that the subtree probability equals the rule probability[1]. In other words, both probability models are based on the same idea for lexicalization, but the type of the corpora they are estimated from differ (*trees* versus *sentences*).

In more detail, Table 1 displays all four grammar-rule types resulting from the grammar transformation of Carroll and Rooth (1998). The underlying entities from the original CFG are: The starting symbol S (also the starting symbol of the transform), the internal rule C $\longrightarrow$ D$_1$ . . .D$_m$ H D$_{m+1}$ . . .D$_{m+n}$, and the lexical rule C $\longrightarrow w$. From these, the context-free transforms are generated as displayed in the table (for all possible head words h and d, and for all non-head children D=D$_1$, . . ., D$_{m+n}$). Figure 4 displays an example parse on the basis of the

[1]at least, if we ignore Charniak's conditioning on C's parent category C$_p$ for the moment; Note that C's parent category is available in the tree-bank, but may not occur in the left-hand sides of the rules of a manually written CFG



$$p_{\text{STANDARD-PCFG}}(\ \textbf{this sub-tree}\ )$$

$$= p(\ \mathbf{D}_1{:}\mathbf{C}{:}h \ldots \mathbf{D}_m{:}\mathbf{C}{:}h\ \ \mathbf{H}{:}h\ \ \mathbf{D}_{m+1}{:}\mathbf{C}{:}h \ldots \mathbf{D}_{m+n}{:}\mathbf{C}{:}\ h\ \mid\ \mathbf{C}{:}h\ )\ \times\ \textstyle\prod_{i=1}^{m+n}\ p(\ \mathbf{D}_i{:}d_i\ \mid\ \mathbf{D}_i{:}\mathbf{C}{:}h\ )$$

$$= p(\ \mathbf{D}_1 \ldots \mathbf{D}_m\ \mathbf{H}\ \mathbf{D}_{m+1} \ldots \mathbf{D}_{m+n}\ \mid\ \mathbf{C}, h\ )\ \times\ \textstyle\prod_{i=1}^{m+n}\ p(\ d_i\ \mid\ \mathbf{D}_i, \mathbf{C}, h\ )$$

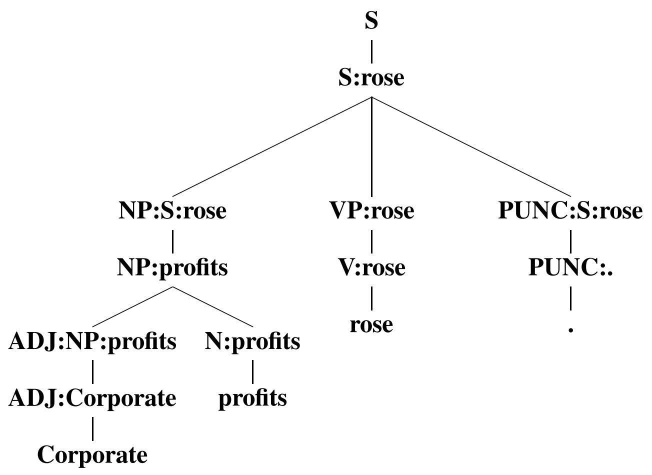
$$= p(\ r\ \mid\ \mathbf{C}, h\ )\ \times\ \textstyle\prod_{i=1}^{m+n}\ p(\ d_i\ \mid\ \mathbf{D}_i, \mathbf{C}, h\ )$$

(r **is the *unlexicalized* rule**)

Figure 3: Transformed internal rule, and its standard-PCFG probability (Carroll and Rooth, 1998)

S	$\longrightarrow$	S:h	**(Starting Rules)**
C:h	$\longrightarrow$	D$_1$:C:h ... D$_m$:C:h H:h D$_{m+1}$:C:h ... D$_{m+n}$:C:h	**(Lexicalized Rules)**
D:C:h	$\longrightarrow$	D:d	**(Dependencies)**
C:w	$\longrightarrow$	w	**(Lexical Rules)**

Table 1: Context-free rule types in the transform (Carroll and Rooth, 1998)



Starting Rule:
S $\longrightarrow$ S:rose
Lexicalized Rules:
S:rose $\longrightarrow$ NP:S:rose VP:rose PUNC:S:rose
NP:profits $\longrightarrow$ ADJ:NP:profits N:profits
VP:rose $\longrightarrow$ V:rose
Dependencies:
NP:S:rose $\longrightarrow$ NP:profits
PUNC:S:rose $\longrightarrow$ PUNC:.
ADJ:NP:profits $\longrightarrow$ ADJ:Corporate
Lexical Rules:
ADJ:Corporate $\longrightarrow$ Corporate
N:profits $\longrightarrow$ profits
V:rose $\longrightarrow$ rose
PUNC:. $\longrightarrow$.

Figure 4: Transformed parse tree, and a list of the rules it contains (Carroll and Rooth, 1998)

transformed grammar. It is noteworthy that although Carroll and Rooth (1998) learn from a text corpus of about 50 million words, it is still necessary to smooth the rule probabilities of the transform. Unlike Charniak (1997), however, they do not use word classes in their back-off scheme.

To summarize, the major problem of full-lexicalization techniques is that they lead to serious sparse-data problems. For both models presented in this section, a large number $|T|$ of full word forms makes it difficult to reliably estimate the probability weights of the $O(|T|^2)$ dependencies and the $O(|T|)$ lexicalized rules.

A linguistically naive approach to this problem is to use POS tags as heads to decrease the number of heads. From a computational perspective, the sparse data problem would then be completely solved since the number $|\text{POS}|$ of POS tags is tiny compared to the number $|T|$ of full-word forms. Although we will demonstrate that parsing results benefit already from this naive lexicalization routine, we expect that (computationally and linguistically) optimal head-lexicalized models are arranged around a number $|\text{HEADS}|$ of head elements such that $|\text{POS}| \leq |\text{HEADS}| \ll |T|$.

3 Latent-Head Models

This section defines two probability models over the trees licensed by a head-lexicalized CFG with latent head-information, thereby exploiting three simple linguistic principles: (i) all rules have head markers, (ii) information is projected up a chain of categories marked as heads, (iii) lexical entries carry latent head values which can be learned. Moreover, two estimation methods for the latent-head models are described.

Head-Lexicalized CFGs with Latent Heads

Principles (i) and (ii) are satisfied by all head lexicalized models we know of, and clearly, they are also satisfied by the model of Carroll and Rooth (1998). Principle (iii), however, deals with latent information for lexical entries, which is beyond the capability of this model. To see this, remember that lexical rules C $\longrightarrow w$ are unambiguously transformed to C:$w \longrightarrow w$. Because this transformation is unambiguous, latent information does not play a role in it. It is surprisingly simple, however, to satisfy principle (iii) with slightly modified versions of Carroll and Rooth's transformation of lexical rules. In the following, we present two of them:

Lexical-Rule Transformation (Model 1): Transform each lexical rule C $\longrightarrow w$ to a set of rules, having the form C:$h \longrightarrow w$, where $h \in \{1, \ldots, L\}$, and L is a free parameter.

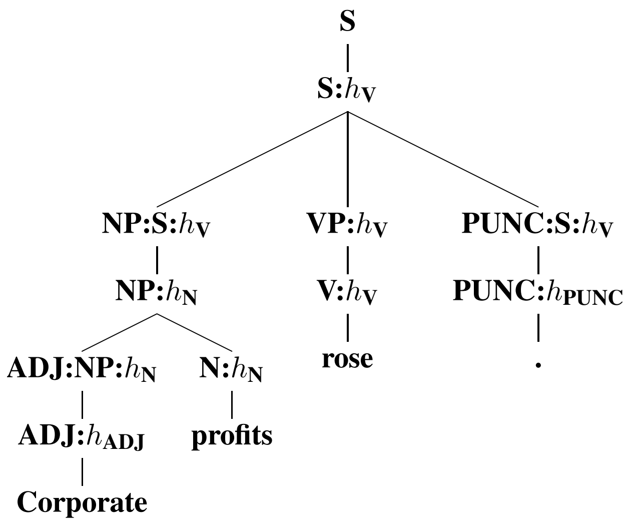
Lexical-Rule Transformation (Model 2): Transform each lexical rule C $\longrightarrow w$ to a set of rules, having the form C:$h \longrightarrow w$, where $h \in \{\text{C}\} \times \{1, \ldots, L\}$, and L is a free parameter.

Both models introduce latent heads for lexical entries. The difference is that Model 1 introduces completely latent heads h, whereas Model 2 introduces heads h on the basis of the POS tag C of the word w: each such head is a combination of C with an abstract extra-information. Figure 5 gives an example. Because we still apply Carroll and Rooth's grammar transformation scheme to the non-lexical rules, latent heads are percolated up a path of categories marked as heads.

Although our modifications are small, their effect is remarkable. In contrast to Carroll and Rooth (1998), where an unlexicalized tree is unambiguously mapped to a *single* transform, our models map an unlexicalized tree to *multiple* transforms (for free parameters ≥ 2). Note also that although latent information is freely introduced at the lexical level, it is not freely distributed over the nodes of the tree. Rather, the space of latent heads for a tree is constrained according the linguistic principle of headedness. Finally, for the case $L = 1$, our models perform unambiguous transformations: in Model 1 the transformation makes no relevant changes, whereas Model 2 performs unambiguous lexicalization with POS tags. In the rest of the paper, we show how to learn models with hidden, richer, and more accurate head-information from a tree-bank, if $L \geq 2$.

Unsupervised Estimation of Head-Lexicalized CFGs with Latent Heads

In the following, we define two methods for estimating latent-head models. The main difficulty here is that the rules of a head-lexicalized CFG



Starting Rule:

S $\longrightarrow$ S:h_{V}

Lexicalized Rules:

S:h_{V} $\longrightarrow$ NP:S:h_{V} VP:h_{V} PUNC:S:h_{V}

NP:h_{N} $\longrightarrow$ ADJ:NP:h_{N} N:h_{N}

VP:h_{V} $\longrightarrow$ V:h_{V}

Dependencies:

NP:S:h_{V} $\longrightarrow$ NP:h_{N}

PUNC:S:h_{V} $\longrightarrow$ PUNC:h_{PUNC}

ADJ:NP:h_{N} $\longrightarrow$ ADJ:h_{ADJ}

Lexical Rules:

ADJ:h_{ADJ} $\longrightarrow$ Corporate

N:h_{N} $\longrightarrow$ profits

V:h_{V} $\longrightarrow$ rose

PUNC:h_{PUNC} $\longrightarrow$.

Model 1 (Completely Latent Heads):

h_{ADJ}, h_{N}, h_{V}, and $h_{\mathrm{PUNC}} \in \{1, \ldots, L\}$

Model 2 (Latent Heads Based on POS Tags):

$h_{\mathrm{ADJ}} \in \{\mathrm{ADJ}\} \times \{1, \ldots, L\}$

$h_{\mathrm{N}} \in \{\mathrm{N}\} \times \{1, \ldots, L\}$

$h_{\mathrm{V}} \in \{\mathrm{V}\} \times \{1, \ldots, L\}$

$h_{\mathrm{PUNC}} \in \{\mathrm{PUNC}\} \times \{1, \ldots, L\}$

$$\textbf{Number of Latent-Head Types} = \begin{cases} L & \text{for Model 1} \\ |POS| \times L & \text{for Model 2} \end{cases} \qquad (L \text{ is a free parameter})$$

Figure 5: Parse tree with latent heads, and a list of the rules it contains.

```
Initialization: Generate a randomly initialized distribution $p_0$ for the rules of $G_{\text{LEX}}$ (a head-
lexicalized CFG with latent heads as previously defined).

Iterations:
(1) for each $i = 1, 2, 3, \ldots$, number_of_iterations do
(2)     set $p = p_{i-1}$
(3)     E step: Generate a lexicalized tree-bank $T_{\text{LEX}}$, by
          - running over all unlexicalized trees $t$ of the original tree-bank
          - generating the finite set $G_{\text{LEX}}(t)$ of the lexicalized transforms of $t$
          - allocating the frequency $c(t') = c(t) \cdot p(\, t' \mid t \,)$ to the lexicalized trees $t' \in G_{\text{LEX}}(t)$
                                        [ Here, $c(t)$ is the frequency of $t$ in the original tree-bank ]
(4)     M step: Read the tree-bank grammar off $T_{\text{LEX}}$, by
          - calculating relative frequencies $\hat{p}$ for all rules of $G_{\text{LEX}}$ as occurring in $T_{\text{LEX}}$
(5)     set $p_i = \hat{p}$
(6) end
```

Figure 6: Grammar induction algorithm (EM algorithm)

with latent heads cannot be directly estimated from the tree-bank (by counting rules) since the latent heads are not annotated in the trees. Faced with this incomplete-data problem, we apply the Expectation-Maximization (EM) algorithm developed for these type of problems (Dempster et al., 1977). For details of the EM algorithm, we refer to the numerous tutorials on EM (e.g. Prescher (2003)). Here, it suffices to know that it is a sort of meta algorithm, resulting for each incomplete-data problem in an iterative estimation method that aims at maximum-likelihood estimation on the data. Disregarding the fact that we implement a dynamic-programming version for our experiments (running in linear time in the size of the trees in the tree-bank (Prescher, 2005)), the EM algorithm is here as displayed in Figure 6. Beside this pure form of the EM algorithm, we also use a variant where the original tree-bank is annotated with most probable heads only. Here is a characterization of both estimation methods:

Estimation from latent-head distributions: The key steps of the EM algorithm produce a lexicalized tree-bank T_{LEX}, consisting of all lexicalized versions of the original trees (E-step), and calculate the probabilities for the rules of G_{LEX} on the basis of T_{LEX} (M-step). Clearly, all lexicalized trees in $G_{\text{LEX}}(t)$ differ only in the heads of their nodes. Thus, EM estimation uses the original tree-bank, where each node can be thought of as annotated with a *latent-head distribution*.

Estimation from most probable heads: By contrast, a quite different scheme is applied in Klein and Manning (2003): extensive manual annotation enriches the tree-bank with information, but no trees are added to the tree-bank. We borrow from this scheme in that we take the best EM model to calculate the most probable head-lexicalized versions of the trees in the original tree-bank. After collecting this Viterbi-style lexicalized tree-bank, the ordinary tree-bank estimation yields another estimate of G_{LEX}. Clearly, this estimation method uses the original tree-bank, where each node can be thought of annotated with the *most probable latent head*.

4 Experiments

This section presents empirical results across our models and estimation methods.

Data and Parameters

To facilitate comparison with previous work, we trained our models on sections 2-21 of the WSJ section of the Penn tree-bank (Marcus et al., 1993). All trees were modified such that: The empty top node got the category TOP, node labels consisted solely of syntactic category information, empty nodes (i.e. nodes dominating the empty string) were deleted, and words in rules occurring less than 3 times in the tree-bank were replaced by (word-suffix based)

	Estimation from most probable heads				Estimation from head distributions			
	Model 1 (completely latent)		**Model 2** (POS+latent)		**Model 1** (completely latent)		**Model 2** (POS+latent)	
baseline	(15 400)	73.5	(25 000)	78.9	(15 400)	73.5	(25 000)	78.9
L=2	(17 900)	76.3	(32 300)	81.1	(25 900)	76.9	(49 500)	81.6
L=5	(22 800)	80.7	(46 200)	*83.3*	(49 200)	82.0	(116 300)	84.9
L=10	(28 100)	*83.3*	(58 900)	82.6	(79 200)	*84.6*	(224 300)	**85.7**
		Δ=9.8		Δ=4.4		Δ=**11.1**		Δ=6.8

Table 2: Parsing results in LP/LR F_1 (the baseline is $L = 1$)

unknown-word symbols. No other changes were made.

On this tree-bank, we trained several head-lexicalized CFGs with latent-heads as described in Section 3, but smoothed the grammar rules using deleted interpolation; We also performed some preliminary experiments without smoothing, but after observing that about 3000 trees of our training corpus were allocated a zero-probability (resulting from the fact that too many grammar rules got a zero-probability), we decided to smooth all rule probabilities.

We tried to find optimal starting parameters by repeating the whole training process multiple times, but we observed that starting parameters affect final results only up to 0.5%. We also tried to find optimal iteration numbers by evaluating our models after each iteration step on a held-out corpus, and observed that the best results were obtained with 70 to 130 iterations. Within a wide range from 50 to 200 iteration, however, iteration numbers affect final results only up to 0.5%

Empirical Results

We evaluated on a parsing task performed on Section 22 of the WSJ section of the Penn tree-bank. For parsing, we mapped all unknown words to unknown word symbols, and applied the Viterbi algorithm as implemented in Schmid (2004), exploiting its ability to deal with highly-ambiguous grammars. That is, we did not use any pruning or smoothing routines for parsing sentences. We then de-transformed the resulting maximum-probability parses to the format described in the previous sub-section. That is, we deleted the heads, the dependencies, and the starting rules. All grammars were able to exhaustively parse the evaluation corpus. Table 2 displays our results in terms of LP/LR F_1 (Black and al., 1991). The largest number per column is printed in italics. The absolutely largest number is printed in boldface. The numbers in brackets are the number of grammar rules (without counting lexical rules). The gain in LP/LR F_1 per estimation method and per model is also displayed (Δ). Finally, the average training time per iteration ranges from 2 to 4 hours (depending on both L and the type of the model). The average parsing time is 10 seconds per sentence, which is comparable to what is reported in Klein and Manning (2003).

5 Discussion

First of all, all model instances outperform the baseline, i.e., the original grammar (F_1=73.5), and the head-lexicalized grammar with POS tags as heads (F_1=78.9). The only plausible explanation for these significant improvements is that useful head classes have been learned by our method. Moreover, increasing L consistently increases F_1 (except for Model 2 estimated from most probable heads; $L = 10$ is out of the row). We thus argue that the granularity of the current head classes is not fine enough; Further refinement may lead to even better latent-head statistics.

Second, estimation from head distributions consistently outperforms estimation from most probable heads (for both models). Although coarse-grained models clearly benefit from POS information in the heads ($L = 1, 2, 5$), it is surprising that the *best* models with completely latent heads are on a par with or almost as good as the *best* ones using POS

	LP	LR	F_1	Exact	CB
Model 1 (this paper)	84.8	84.4	84.6	26.4	1.37
Magerman (1995)	84.9	84.6			1.26
Model 2 (this paper)	85.7	85.7	85.7	29.3	1.29
Collins (1996)	86.3	85.8			1.14
Matsuzaki etal. (2005)	86.6	86.7			1.19
Klein and Manning (2003)	86.9	85.7	86.3	30.9	1.10
Charniak (1997)	87.4	87.5			1.00
Collins (1997)	88.6	88.1			0.91

Table 3: Comparison with other parsers (sentences of length ≤ 40)

as head information.

Finally, our absolutely best model (F_1=85.7) combines POS tags with latent extra-information ($L = 10$) and is estimated from latent-head distributions. Although it also has the largest number of grammar rules (about 224 300), it is still much smaller than fully-lexicalized models. The best model with completely latent heads, however, leads to almost the same performance (F_1=84.6), and has the further advantage of having significantly fewer rules (only about 79 200). Moreover, it is the model which leads to the largest gain compared to the baseline ($\Delta = 11.1$).

In the rest of the section, we compare our method to related methods. To start with performance values, Table 3 displays previous results on parsing Section 23 of the WSJ section of the Penn tree-bank. Comparison indicates that our best model is already better than the early lexicalized model of Magerman (1995). It is a bit worse than the unlexicalized PCFGs of Klein and Manning (2003) and Matsuzaki et al. (2005), and of course, it is also worse than state-of-the-art lexicalized parsers (experience shows that evaluation results on sections 22 and 23 do not differ much).

Beyond performance values, we believe our formalism and methodology have the following attractive features: first, our models incorporate context and lexical information collected from the whole tree-bank. Information is bundled into abstract heads of higher-order information, which results in a drastically reduced parameter space. In terms of Section 2, our approach does not aim at improving the approximation of rule probabilities $p(r|\mathrm{C}, h)$ and dependency probabilities $p(d|\mathrm{D}, \mathrm{C}, h)$ by smoothing. Rather, our approach induces head classes for the words h and d from the tree-bank and aims at a exact calculation of rule probabilities $p(r|\mathrm{C}, \mathrm{class}(h))$ and dependency probabilities $p(\mathrm{class}(d)|\mathrm{D}, \mathrm{C}, \mathrm{class}(h))$. This is in sharp contrast to the smoothed fixed-word statistics in most lexicalized parsing models derived from sparse data (Magerman (1995), Collins (1996), Charniak (1997), etc.). Particularly, class-based dependency probabilities $p(\mathrm{class}(d)|\mathrm{D}, \mathrm{C}, \mathrm{class}(h))$ induced from the tree-bank are not exploited by most of these parsers.

Second, our method results in an *automatic* linguistic mark-up of tree-bank grammars. In contrast, manual linguistic mark-up of the tree-bank like in Klein and Manning (2003) is based on individual linguistic intuition and might be cost and time intensive.

Third, our method can be thought of as a new lexicalization scheme of CFG based on the notion of latent head-information, or as a successful attempt to incorporate lexical classes into parsers, combined with a new word clustering method based on the context represented by tree structure. It thus complements and extends the approach of Chiang and Bikel (2002), who aim at discovering latent head *markers* in tree-banks to improve manually written head-percolation rules.

Finally, the method can also be viewed as an extension of *factorial HMMs* (Ghahramani and Jordan, 1995) to PCFGs: the node labels on trees are enriched with a latent variable and the latent variables are learned by EM. Matsuzaki et al. (2005) independently introduce a similar approach and present empirical results that rival ours. In contrast to us, they do not use an *explicit linguistic grammar*, and they do not attempt to *constrain* the space of latent variables *by linguistic principles*. As a consequence, our best models are three orders of magnitude more space efficient than theirs (with about 30 000 000 parameters). Therefore, parsing with their models requires sophisticated smoothing and pruning, whereas parsing with ours does not. Moreover, we calculate the most probable latent-head-decorated parse and delete the latent heads in a post-processing step. This is comparable to what they call 'Viterbi complete tree' parsing. Under this regime, our parser is on a par with theirs (F_1=85.5). This suggests that both models have learned a comparable degree of information, which is surprising, because we learn latent heads only, whereas they aim at learning general features. Crucially, a final 1% improvement comes from selecting most-probable parses by bagging all complete parses with the same incomplete skeleton beforehand; Clearly, a solution to this NP-Complete problem (Sima'an, 2002) can/should be also incorporated into our parser.

6 Conclusion

We introduced a method for inducing a head-driven PCFG with latent-head statistics from a tree-bank. The automatically trained parser is time and space efficient and achieves a performance already better than early lexicalized ones. This result suggests that our grammar-induction method can be successfully applied across domains, languages, and tree-bank annotations.

Acknowledgment

This work was supported by the Netherlands Organization for Scientific Research, NWO project no. 612.000.312, 'Learning Stochastic Tree-Grammars from Tree-banks'. I also would like to thank Yoav Seginer and Jelle Zuidema and the anonymous reviewers. A special thanks goes to Khalil Sima'an.

References

Ezra Black and al. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proc. of DARPA-91*.

Joan Bresnan and Ronald M. Kaplan. 1982. Lexical functional grammar: A formal system for grammatical representation. In *The Mental Representation of Grammatical Relations*. MIT Press.

Glenn Carroll and Mats Rooth. 1998. Valence induction with a head-lexicalized PCFG. In *Proc. of EMNLP-3*.

Eugene Charniak. 1997. Parsing with a context-free grammar and word statistics. In *Proc. of AAAI-97*.

David Chiang and D. Bikel. 2002. Recovering latent information in treebanks. In *Proc. of COLING-02*.

Michael Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proc. of ACL-96*.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of ACL-97*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *J. Royal Statist. Soc.*, 39(B).

Amit Dubey and Frank Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *Proc. of ACL-03*.

Zoubin Ghahramani and Michael Jordan. 1995. Factorial Hidden Markov Models. Technical report, MIT.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL-03*.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proc. of ACL-95*.

Mitch Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2).

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proc. of ACL-05*.

Detlef Prescher. 2003. A Tutorial on the Expectation-Maximization Algorithm Including Maximum-Likelihood Estimation and EM Training of Probabilistic Context-Free Grammars. Presented at the *15th European Summer School in Logic, Language and Information (ESSLLI)*.

Detlef Prescher. 2005. Inducing Head-Driven PCFGs with Latent Heads: Refining a Tree-bank Grammar for Parsing. In *Proc. of the 16th European Conference on Machine Learning*.

Helmut Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proc. of COLING-04*.

Khalil Sima'an. 2002. Computational complexity of probabilistic disambiguation. *Grammars*, 5(2).