Inducing Head-Driven PCFGs with Latent Heads: Refining a Tree-Bank Grammar for Parsing

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 a probabilistic parser with latent head information from simple linguistic principles. Our parser has a performance of 85.1% (LP/LR F_1), which is as good as that of early *lexicalized* ones. This is remarkable since the induction of probabilistic grammars is in general a hard task.

1 Introduction

State-of-the-art statistical parsers for natural language are based on probabilistic grammars acquired from tree-banks. The method of acquiring a probabilistic grammar from the tree-bank is of major influence on the accuracy and coverage of the statistical parser. It turns out that directly acquiring the probabilistic grammar from the tree-bank results in a suboptimal statistical parser [1]. Thus, various linguistically motivated transformation techniques have been applied to the tree-bank trees, all of them gathering important local information at the context-free production level. Two major transforms are currently used in the literature: *Parent encoding* and *lexicalization* ([2], [3], [4], [5], [6], etc.). Parent encoding appends the parent label to the tree nodes. Lexicalization labels every node with the head word. These transforms have been specifically developed for English based on the linguistic intuition that the original tree-bank annotations are not refined enough to capture the various lexical and contextual influences that could improve parser performance. It turns out, however, that these transforms do not carry over across different tree-banks for other languages, annotations or domains ([7], [8]), and even parsing English relies on some sophisticated further refinements [9]. Finally, all lexicalized models we are aware of have to incorporate smoothing and pruning techniques to solve a serious sparse-data problem (cp. Section 2).

Recently, [10] showed that a carefully performed *linguistic mark-up* leads to almost the same performance results as lexicalization (both combined with parent encoding). This result is attractive since unlexicalized grammars are easy to estimate, easy to parse with, and time- and space-efficient. Furthermore, linguistic annotations orthogonal to

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 292–304, 2005.

lexicalization could presumably be used to benefit lexicalized parsers as well. A drawback of [10]'s method is, however, that their 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 for languages other than English.

In this context, it is thus important to answer the following question: Is it possible to *automatically induce* a more refined probabilistic grammar from a given tree-bank with improved performance? Our answer is yes, and the resulting parser is located in the middle of 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 greater detail, our induction method uses the same linguistic principles of headedness as other methods: We do believe that lexical information represents an important knowledge source. Simply percolating lexical information including the words, however, leads to data sparseness. Various advanced linguistic theories (e.g. Lexical-Functional Grammar [11]) suggest that more abstract categories based on feature combinations could represent the lexical effect. Our main assumption is based on these theories but complemented by a learning paradigm: Lexical entries carry *latent* extra information, and the *combinations of POS tags and extra-information* serve as partly hidden head elements of a probabilistic grammar to be induced from the tree-bank. It is important to emphasize that our task is to *automatically induce* a more refined probabilistic grammar based on a few linguistic principles. With *automatic refinement* it is harder to guarantee improved performance than with manually tailored refinements [10] or with refinements based on direct lexicalization [6]. However, if the induced refinement provides improved performance then it has a clear advantage: it is automatically induced, which gives the hope that it will be applicable across different domains, languages and tree-bank annotations.

In this paper we study the utility of a well-known statistical learning algorithm, the Expectation-Maximization (EM) algorithm [12] for the refinement of probabilistic grammars. Because we work with Probabilistic Context-Free Grammars (PCFGs), we specifically employ the Inside-Outside version of the EM. Applying our method to the benchmark Penn tree-bank Wall-Street Journal (WSJ), 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 with lexicalized probabilistic grammars that were obtained using a direct transform. This is a remarkable result given the hard task of automatic induction of improved probabilistic grammars.

2 Head Lexicalization

As previously shown ([2], [3], [4], 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 the rules is used to project lexical items up a chain of nodes. One of the simplest approaches to lexicalization is the one of [13]. It is characterized by the following transformation of the original tree-bank CFG.

Definition. The set T of terminal symbols of the original CFG and of its transform are identical. The non-terminal symbols of the transform have the form $X[v]$ or $\lhd Y \lhd X[v] \rhd \rhd$.

Here, X and Y are two arbitrary non-terminal symbols of the original CFG, and v is an arbitrary head chosen from a finite set $\mathcal{H}$ of head symbols. The set of rules of the transformed CFG consists of the following types:

Lexicalized starting rules: For all heads $v \in \mathcal{H}$ (where ROOT is a new start symbol and S is the original one):

$$\text{ROOT} \rightarrow S[v]$$

Lexicalized rules: For all heads $v \in \mathcal{H}$ and for all rules $X \rightarrow \ldots X_{i-1}\ X_i\ X_{i+1}\ \ldots$ of the original CFG (with a head marker on the child X_i):

$$X[v] \rightarrow \ldots <X_{i-1}<X[v]\gg\ X_i[v]\ <X_{i+1}<X[v]\gg\ \ldots$$

Lexicalized grammatical relations: For all new non-terminal symbols $<Y<X[v]\gg$ and for all heads $w \in \mathcal{H}$:

$$<Y<X[v]\gg \rightarrow Y[w]$$

Lexical rules: For all lexical rules $X \rightarrow w$ of the original CFG:

$$X[h(w)] \rightarrow w$$

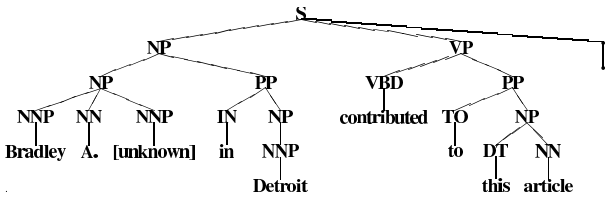

Fig. 1. Original context-free tree (Note: One word was replaced earlier by the unknown-word symbol '[unknown]').

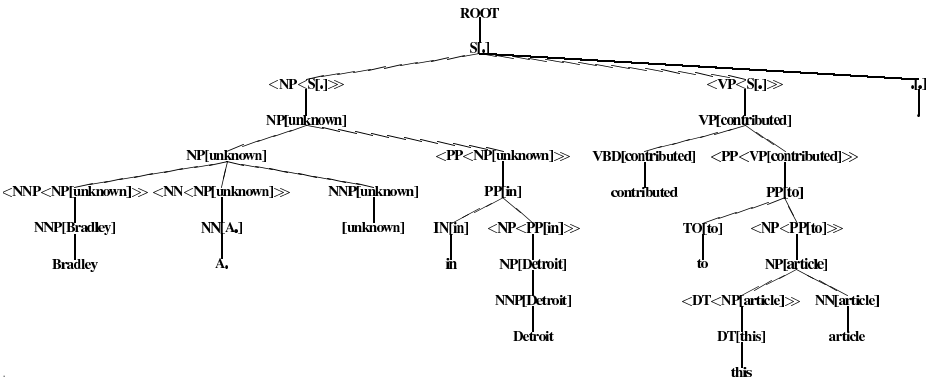

Fig. 2. Transformed tree: Lexicalization with leaf nodes of the original tree. The nodes $<cat_1<cat_2[head]\gg$ are auxiliary nodes which have been introduced to model the lexicalization of rules independently from the lexicalization of grammatical relations [13]. Additionally, auxiliary nodes reduce the sparse-data problem.

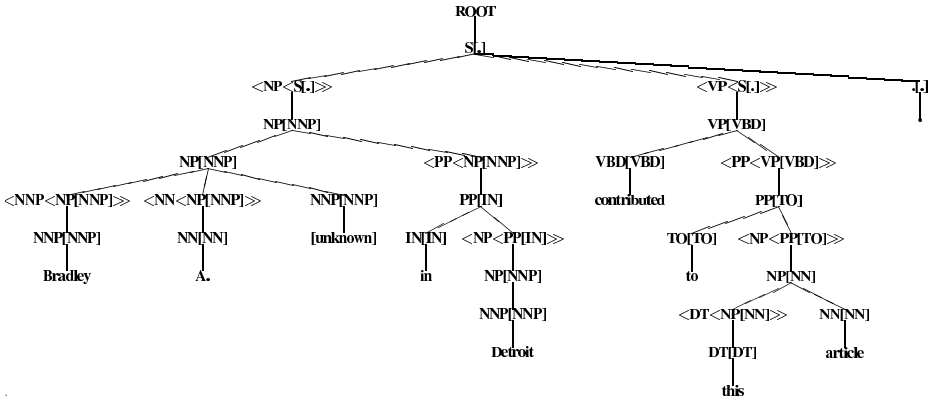

Fig. 3. Transformed tree: The same model but with lexicalization performed with POS tags

Here, $h : V \rightarrow \mathcal{H}$ is a many-to-one function mapping a non-terminal $w \in V$ to its head symbol $h(w) \in \mathcal{H}$ (e.g. to the word itself or to the lemma of the word). Figure 2 displays the result of this transformation for one of the trees of the training section of the Penn tree-bank. Compared to the original tree in Figure 1, it is note worthy that the lexicalized tree has more nodes than the original tree: for each non-head daughter cat_1 of a mother node cat_2 in the original tree, a new node $<cat_1<cat_2[head]>>$ is introduced in the lexicalized tree. We call these extra nodes **auxiliary nodes**. The main reason for introducing them is simply that one does not want to lexicalize rules with more than one head (otherwise, it would be difficult to overcome the arising sparse-data problem). Note also that standard-probabilistic conditioning of the lexicalized rules results simply in conditioning the *un*lexicalized rules on a lexical head, such as in:

$$
\begin{aligned}
&p(\text{ NP[article]} \rightarrow \text{<DT<NP[article]>> NN[article] | NP[article] }) \\
&= \\
&p(\text{ NP} \rightarrow \text{DT NN | NP, article })
\end{aligned}
$$

One problem of head-lexicalization techniques is that they lead to serious sparse data problems. For the standard case $h(w) = w$, for example, the large number $|T|$ of full word forms makes it difficult to reliably estimate the probability weights of the $O(|T|^2)$ lexicalized grammatical relations and $O(|T|)$ lexicalized rules of the model of [13]. An obvious approach to the problem is to use lemmas instead of full word forms to decrease the number of heads. From a computational perspective (but of course not from the linguistic one) the sparse data problem can be solved if part-of-speech tags are used as heads since the number of POS tags is tiny compared to $|T|$. Figure 3 displays the result of this type of transformation. Although we will demonstrate that parsing results benefit from this naive lexicalization routine, we expect that (computationally and linguistically) optimal head-lexicalized models are arranged around a number $|\mathcal{H}|$ of head symbols such that $|POS| \leq |\mathcal{H}| \ll |T|$, where POS is the set of POS tags and T is the full-word-form lexicon.

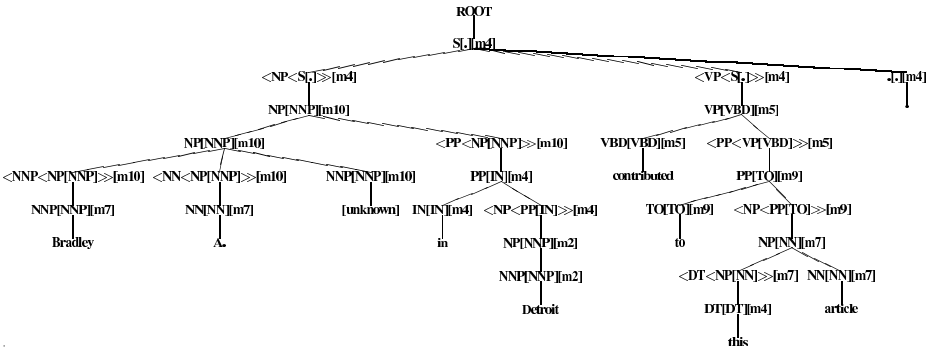

Fig. 4. Head-lexicalized tree with hidden information mark-up: In addition to the part-of-speech information, the terminal nodes carry extra-information (like m7, m10, m4, etc.). This satisfies principle (iii). The possible combinations of part-of-speech tags and extra-information (e.g., [NNP][m7], [NNP][m10], [IN][m4], etc.) serve as the new head elements of a head-lexicalized CFG, because the combined information is projected up a chain of categories. This satisfies principles (i) and (ii). The extra-information of the lexical items is hidden since it is not annotated in the Penn tree-bank. Therefore, the information mark-up is highly ambiguous, and there are many more (differently marked-up) trees beside the displayed tree.

3 Modeling Hidden Head-Information

This section defines probability models over the trees licensed by a lexicalized CFG with hidden 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 hidden extra-information which can be revealed.

Principles (i) and (ii) are satisfied by all head lexicalization routines which we know of. We base our model on the relatively simple head-lexicalized model presented in Section 2 because we do not want to explore **how** hidden extra-information flows in a tree-bank. Rather we would like to induce **which** extra-information flows in single trees and in tree-banks. Figure 4 displays a simple example of the type of extra-information mark-up we are interested in. Compared to the tree in Figure 3, all nodes carry some abstract extra-information type of the form $m_1, m_2, m_3, ...$ These information types are introduced at the part-of-speech level, and the combination of POS tag and extra-information flows bottom-up along a chain of categories marked as heads. In other words, the combination of POS tags and extra-information forms new but more complex heads of the head-lexicalized CFG. Moreover, we explicitly allow for ambiguous heads in the lexical rules. Formally, the mark-up of extra-information and the flow of the combination of original heads and extra-information can be done via the following transformation of the head-lexicalized CFG introduced in Section 2.

Definition. The set T of terminal symbols and the set $\mathcal{H}$ of head symbols remain unchanged. The non-terminal symbols of the transform have the form $X[v][m]$ or

$\lessdot Y \lessdot X[v] \gtrdot$[m] . Here, $X[v]$ and $\lessdot Y \lessdot X[v] \gtrdot$ are non-terminals of the original head-lexicalized CFG, while m is an extra-information type chosen from a finite set $\mathcal{I}$. (Throughout this paper, we use $\mathcal{I} = \{m_1 \dots m_n\}$.) The set of rules of the transform consists of the following types:

Lexicalized starting rules with info mark-up: For all extra-information types $m \in \mathcal{I}$ and for all heads $v \in \mathcal{H}$ (where TOP is a new start symbol):

$$\text{TOP} \rightarrow S[v][m]$$

Lexicalized rules with info mark-up: For all extra-information types $m \in \mathcal{I}$ and for all rules $X[v] \rightarrow \dots \lessdot X_{i-1} \lessdot X[v] \gtrdot X_i[v] \lessdot X_{i+1} \lessdot X[v] \gtrdot \dots$ of the head-lexicalized CFG:

$$X[v][m] \rightarrow \dots \lessdot X_{i-1} \lessdot X[v] \gtrdot [m]\ X_i[v][m] \lessdot X_{i+1} \lessdot X[v] \gtrdot [m] \dots$$

Lexicalized grammatical relations with info mark-up: For all pairs of extra-information types $i, j \in \mathcal{I}$ and for all rules $\lessdot Y \lessdot X[v] \gtrdot \rightarrow Y[w]$ of the head-lexicalized CFG:

$$\lessdot Y \lessdot X[v] \gtrdot [i] \rightarrow Y[w][j]$$

Lexical rules with info mark-up: For all extra-information types $m \in \mathcal{I}$ and for all lexical rules $X[h(w)] \rightarrow w$ of the head-lexicalized CFG:

$$X[h(w)][m] \rightarrow w$$

Thus, a head-lexicalized CFG with unambiguous extra-information mark-up contains exactly the same information as the original head-lexicalized CFG. In the rest of the paper, we show, however, that it is possible to learn hidden, richer, and more accurate head information from tree-banks.

4 Estimating Hidden Head-Information

Given a head-lexicalized CFG, the inductive problem is to estimate a head-lexicalized CFG with extra-information mark-up. The difficulty is that the rules of the marked-up CFG can not be directly estimated from the Penn tree-bank (by counting rules) because the extra-information mark-up is not annotated in the tree-bank. Therefore, we work with the standard method for unsupervised estimation of PCFGs, the inside-outside algorithm [14]. This algorithm induces probabilities for the grammar rules from a corpus of sentences. To exploit all linguistic information provided by the given tree-bank, we have to use *trees* as *input sentences* for the IO algorithm.

We thus create a context-free grammar which takes a whole head-lexicalized tree as input (see Figure 3) and which outputs the same tree marked-up with extra-information (see Figure 4). We call this grammar a **tree-transformation grammar**, as both its input and its output are trees. The tree-transformation grammar is characterized by the following transformation of the head-lexicalized CFG introduced in Section 2.

Definition. The set of terminal symbols of the transform comprises all symbols occurring in the bracket notations of the input trees, i.e., it consists of both terminal

and non-terminal symbols of the head-lexicalized CFG, as well as of two bracket-symbols '(' and ')'. The non-terminal symbols of the transform have the form $X[v][m]$ or $\lhd Y \lhd X[v] \rhd\!\rhd [\mathrm{m}]$. Here, $X[v]$ and $\lhd Y \lhd X[v] \rhd\!\rhd$ are non-terminals of the original head-lexicalized CFG, and m is an extra-information type chosen from a finite set $\mathcal{I}$. The set of rules of the transform consists of the following types:

Lexicalized starting rules with info mark-up: For all extra-information types $m \in \mathcal{I}$ and for all heads $v \in \mathcal{H}$ (where TOP is a new start symbol):

$$\text{TOP} \rightarrow (\ \text{ROOT}\ \ S[v][m]\ )$$

Lexicalized rules with info mark-up: For all extra-information types $m \in \mathcal{I}$ and for all rules $X[v] \rightarrow \ldots \lhd X_{i-1} \lhd X[v] \rhd\!\rhd\ X_i[v]\ \lhd X_{i+1} \lhd X[v] \rhd\!\rhd\ \ldots$ of the head-lexicalized CFG:

$$X[v][m] \rightarrow (\ X[v]\ \ \ldots\ \ \lhd X_{i-1} \lhd X[v] \rhd\!\rhd [m]\ X_i[v][m]\ \lhd X_{i+1} \lhd X[v] \rhd\!\rhd [m]\ \ldots)$$

Lexicalized grammatical relations with info mark-up: For all pairs of extra-information types $i, j \in \mathcal{I}$ and for all rules $\lhd Y \lhd X[v] \rhd\!\rhd \rightarrow Y[w]$ of the head-lexicalized CFG:

$$\lhd Y \lhd X[v] \rhd\!\rhd [i] \rightarrow (\ \lhd Y \lhd X[v] \rhd\!\rhd\ Y[w][j]\ )$$

Lexical rules with info mark-up: For all extra-information types $m \in \mathcal{I}$ and for all lexical rules $X[h(w)] \rightarrow w$ of the head-lexicalized CFG:

$$X[h(w)][m] \rightarrow (\ X[h(w)]\ \ w\ )$$

For example, the bracket notation of the tree in Figure 3 is as follows:

```
( ROOT ( S[.] ( <NP<S[.]>> ( NP[NNP] ( NP[NNP] ( <NNP<NP[NNP]>> ( NNP[NNP]
 Bradley ) ) ( <NN<NP[NNP]>> ( NN[NN] A. ) ) ( NNP[NNP] [unknown] ) )
( <PP<NP[NNP]>> ( PP[IN] ( IN[IN] in ) ( <NP<PP[IN]>> ( NP[NNP] ( NNP[NNP]
 Detroit ) ) ) ) ) ) ) ( <VP<S[.]>> ( VP[VBD] ( VBD[VBD] contributed ) (
<PP<VP[VBD]>> ( PP[TO] ( TO[TO] to ) ( <NP<PP[TO]>> ( NP[NN] ( <DT<NP[NN]>>
( DT[DT] this ) ) ( NN[NN] article ) ) ) ) ) ) ) ( .[.] . )  )  )
```

It is easy to check that the tree-transformation grammar is able to parse this term. Moreover, the output for this input tree is a parse forest containing (amongst others) the marked-up tree displayed in Figure 4.

Estimation via de-transformation of the tree-transformation grammar: Comparing the definition in this section with the one in the previous section, it is obvious that there is a one-to-one mapping from the rules of the tree-transformation grammar to the rules of the mark-up grammar. For instance, a rule of the tree-transformation grammar having the form

$$marked\_up\_cat \rightarrow (\ cat\ \ marked\_up\_child_1 \ldots marked\_up\_child_n\ )$$

can be simply de-transformed to the following rule of the mark-up grammar

$$marked\_up\_cat \rightarrow marked\_up\_child_1 \ldots marked\_up\_child_n$$

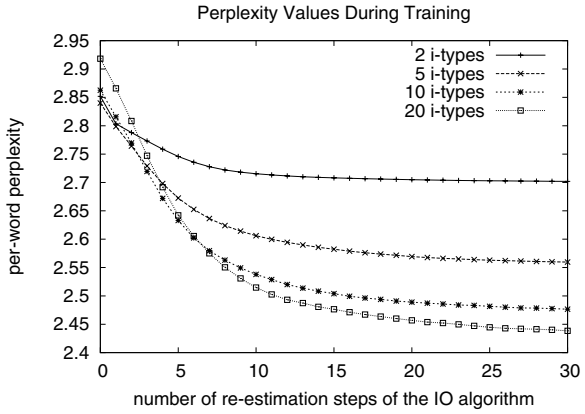

Fig. 5. The plot displays the perplexity of the training corpus for different models at each re-estimation step of the inside-outside algorithm. The displayed perplexity values are per-word-perplexity values as defined in the implementation of [15]. Just as in the standard case, a lower perplexity value corresponds to a higher corpus probability. The models differ in the number of extra-information types for the lexical items. After about 10 iterations, a model with more extra-information types always has a lower perplexity on the training corpus.

In fact, the significant difference between the tree-transformation grammar and the mark-up grammar is that the tree-transformation grammar acts on input trees, whereas the mark-up grammar operates on the yields of these trees. In more detail, the mark-up grammar produces for an *input sentence* the trees of the POS-lexicalized grammar marked-up with all the possible extra-information, whereas the tree-transformation grammar produces the mark-up only for one single *input tree*.

To summarize, the transformation of the mark-up grammar to the tree-transformation grammar enables estimation on the basis of a corpus of trees, whereas the de-transformation of the tree-transformation grammar results in a trained mark-up grammar. Inside-outside estimation of a probabilistic version of the tree-transformation grammar on the tree-bank results thus in a probabilistic version of the mark-up grammar introduced in Section 3. This solves our induction problem.

Implementation (for efficiency improvement): Instead of a single left-bracket symbol '(', we use multiple left-bracket symbols '$(_{id}$' to represent the tree-transformation grammar and its training corpus. The numbers id are identifiers for the rules of the underlying POS-lexicalized CFG. As a consequence, the information mark-up is not estimated in cubic but rather in linear time (in the tree size).

5 Experiments

Using the grammar described in Section 3 and the estimation method described in Section 4, we estimated our models for parts of the Penn tree-bank [16]. To facilitate comparison with previous work, we trained our models on sections 2-21 of the WSJ section

Table 1. Features of the final models: The first column lists the number of extra-information types for the different models. Note that the model with 1 extra-information type is equivalent to lexicalization with POS tags as heads. This model is unambiguous and was not trained with the IO algorithm. The second column lists the total number of rules of the tree-transformation grammars (being used to train our models), thereby only counting the rules with a final non-zero probability. The third column displays the rough number of inside-outside iterations for training, whereas the fourth column lists the rough total training time (resulting from a training time of $2\frac{1}{2} - 4$ hours per iteration step). Finally, the fifth column displays the perplexity of the training corpus for the final models.

i-types	rules	iter	training time	perp
1	53 437	0	0 days	2.821
2	101 385	35	4 days	2.701
5	226 748	35	5 days	2.559
10	396 618	50	7 days	2.467
20	760 894	50	8 days	**2.426**

of the Penn tree-bank. All trees were modified such that: node labels consisted solely of syntactic category information, empty nodes (i.e. nodes dominating the empty string) were deleted, and finally, words in rules occurring less than 3 times in the tree-bank were replaced by an unknown-word symbol '[unknown]'. No other changes were made.

We trained our models with the standard IO algorithm for *un*lexicalized context-free grammars as implemented in [15], thereby activating the built-in (absolute discounting) smoothing routine for grammar rules. We also performed some preliminary experiments *without* smoothing but after observing that about 3000 trees of our training corpus were allocated a zero-probability under IO estimation (resulting from the fact that too many grammar rules got a zero-probability), we decided to smooth all rule probabilities.

Figure 5 displays the training behavior of our models, and Table 1 displays some characteristic features of the final models. After observing that a uniform initialization of the models had no training effect at all, we started the inside-outside algorithm with randomly initialized models. So far, we have not tried to find optimal starting parameters (by repeating the whole training process multiple times), because the current experiment took already months. We also have not tried to find optimal iteration numbers (by evaluating our models after each iteration step on a held-out corpus) because also our evaluation routine is relatively time costly. We therefore simply trained the models until the perplexity values converged. Although our training regime may be sub-optimal (with respect to its fixed starting parameters and the chosen number of iterations), it allows us to systematically investigate models with hundreds of thousands of rules.

6 Evaluation on a Parsing Task

In this section, we evaluate our automatically induced probabilistic grammars on a parsing task. Although parsers developed on the Penn tree-bank are usually evaluated on Section 23 of the WSJ section of the Penn tree-bank, we decided to use Section 22 as evaluation set (sentences with a length $\leq$ 40 only). The reasons for doing this are

Table 2. PARSEVAL scores on our evaluation corpus (Section 22 of the WSJ section of the Penn tree-bank). The columns labeled with LB, LR, F_1, Exact, and CB display values for labeled precision and recall, the harmonic mean, the exact-match rate and the average number of crossing brackets respectively. The table at the top displays the parsing results of our baseline grammar, the original grammar read off slightly modified trees in the training corpus (cp. Figure 1). The larger table displays parsing results for the different models. The first column lists the number of extra-information types for the different models. The model with 1 extra-information type is equivalent to lexicalization with POS tags as heads. It can be regarded as a second baseline. To facilitate comparisons of the PARSEVAL scores with other model features, the second column displays the training-corpus perplexity for the different models. There is a strong correlation with the parsing results: The lower the perplexity the better the parsing result.

Original grammar	LP	LR	F_1	Exact	CB
	75.7	70.1	72.8	10.5	2.14

i-types	perp	LP	LR	F_1	Exact	CB
1	2.821	79.3	77.2	78.2	17.1	1.81
2	2.701	81.6	79.9	80.7	20.1	1.64
5	2.559	84.0	83.2	83.6	25.4	1.44
10	2.467	**85.2**	**85.0**	**85.1**	**27.9**	**1.27**

two-fold. First, most performance figures of [10] refer to parsing results on Section 22 (serving as their development set). Using the same section will facilitate comparison. Second, we envision many extensions and improvements of the present model, and therefore would like to leave Section 23 for future evaluations.

For parsing the sentences of our evaluation corpus, we mapped all unknown words to the unknown word symbol '[unknown]', and applied the Viterbi algorithm as implemented in [17], 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 Section 5. That is, we deleted the extra-information types, the auxiliary nodes, and the POS tags which served as heads. All grammars presented in this section were able to exhaustively parse the evaluation corpus. Table 2 displays our results in terms of the commonly used PARSEVAL scores [18]. The average parsing time in 2GB of memory was 10 seconds per sentence, which is comparable to what is reported in [10].

7 Discussion

In this section, we briefly discuss the experimental results of our final models and compare it to other models. First of all, the size of our models increases almost linear in the number of extra-information types (see Table 1). For instance, the mark-up grammar with 10 extra-information types contains about 400 000 rules, whereas the POS-lexicalized grammar has only about 50 000 rules (i-types=1). The explanation is that the *combinations* of POS tags and extra-information types serve as new abstract head

elements in our models, and therefore, a grammar with x extra-information types contains roughly x-times the number of rules of the POS-lexicalized grammar. However, compared to fully lexicalized grammars, our biggest models are still smaller. Second, the parsing results improve in the number of extra-information types (see Table 2). For instance, modeling with 10 extra-information types results in a F_1 gain of about **12%** compared to the original grammar, and of about **7%** compared to the POS-lexicalized grammar. The only plausible explanation for these significant improvements is that abstract head classes have been learned by our method which are very useful for parsing. Third, it is striking that the difference between the LP and LR scores is almost 6% for the original grammar, about 2% for the POS-lexicalized grammar, and almost 0% for the grammar with 10 extra-information types. In other words, the difference in precision and recall vanishes in the number of extra-information types. We argue that this effect is also related to the fact that useful *classes* of heads have been learned by our models.

In the rest of this section, we compare our method to related methods. To start with performance values, the following table displays previous results on parsing Section 23 of the WSJ section of the Penn tree-bank (sentences of length ≤ 40):

Previous Work	LP	LR	F_1	Exact	CB
Johnson'98			79.7*		
Magerman'95	84.9	84.6			1.26
Collins'96	86.3	85.8			1.14
Klein&Manning'03	86.9	85.7	86.3	30.9	1.10
Charniak'97	87.4	87.5			1.00
Collins'99	88.7	88.6			0.90

Comparison indicates that our best model outperforms parent encoding [5] (*best score of several variants investigated in [10]). It is already as good as the early lexicalized model of [3], a bit worse than the unlexicalized parsing model of [10], and of course 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:

1. The models incorporate context and lexical information collected from the whole tree-bank. Information is bundled into abstract heads of higher-order information. This is in sharp contrast to the fixed-word statistics used in most lexicalized parsing models ([2], [3], [4], [6], etc.) 2. The models have a drastically reduced parameter space compared to lexicalized parsing. Thus they do not suffer from sparse-data problems. 3. The method is based on the original tree-bank and it is not dependent on the success of transformations applied beforehand (like parent-encoding in [6], [10], etc.) 4. The method results in an *automatic* linguistic mark-up of tree-bank grammars. In contrast, manual linguistic mark-up of the tree-bank like in [10] is based on individual linguistic intuition and might be cost and time intensive. 5. The method, we introduced in this paper, can be thought of a new lexicalization scheme of CFG based on the notion of hidden head-information. 6. The method can also be thought of a successful attempt to incorporate lexical classes into parsers, combined with a new word clustering method

based on the context represented in tree structure. 7. It thus complements and extends the approach of [19], which aims at discovering latent head *markers* in tree-banks to improve manually written head-percolation rules. 8. The method is also an extension of *factorial HMMs* [20] to PCFGs: The node labels on trees are enriched with a hidden state and the hidden states are learned with the EM algorithm.

Some of the benefits come at a cost: Clear linguistic interpretation of the induced extra information is currently lacking. It is also possible that extensive manual linguistic mark-up is partly orthogonal to the one we induced. These compromises were made in this paper to answer the important question whether it is possible to *induce* an *accurate* parser from the Penn tree-bank which is not based on full lexicalization.

To conclude, we automatically induced a head-driven PCFG with latent-head statistics from the Penn tree-bank. The resulting parser is as good as early lexicalized parsers. This is a promising result and suggests that our method can be successfully applied across domains, languages, and tree-bank annotations.

Acknowledgment

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

References

1. Charniak, E.: Tree-bank grammars. Technical Report CS-96-02, Brown University (1996)
2. Charniak, E.: Parsing with context-free grammars and word statistics. Technical Report CS-95-28, Department of Computer Science, Brown University (1995)
3. Magerman, D.M.: Statistical decision-tree models for parsing. In: Proc. of ACL'95. (1995)
4. Collins, M.: A new statistical parser based on bigram lexical dependencies. In: Proc. of the ACL'96. (1996)
5. Johnson, M.: PCFG models of linguistic tree representations. Comp. Linguistics **24** (1998)
6. Collins, M.: Head-Driven Statistical Models for Natural Language Parsing. PhD thesis, U of Pennsylvania (1999)
7. Dubey, A., Keller, F.: Probabilistic parsing for German using sister-head dependencies. In: Proc. of ACL'03. (2003)
8. Fissaha, S., Olejnik, D., Kornberger, R., Müller, K., Prescher, D.: Experiments in German treebank parsing. In: Proc. of TSD-03. (2003)
9. Bikel, D.: Intricacies of Collins' parsing model. Computational Linguistics (to appear)
10. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: Proc. of ACL-03. (2003)
11. Bresnan, J., Kaplan, R.M.: Lexical functional grammar: A formal system for grammatical representation. In: The Mental Representation of Grammatical Relations. MIT Press (1982)
12. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the *EM* algorithm. J. Royal Statist. Soc. **39** (1977)
13. Carroll, G., Rooth, M.: Valence induction with a head-lexicalized PCFG. In: Proc. of EMNLP-3. (1998)
14. Lari, K., Young, S.J.: The estimation of stochastic context-free grammars using the inside-outside algorithm. Computer Speech and Language **4** (1990)

15. Schmid, H.: LoPar. Design and Implementation. Technical report, IMS, U Stuttgart (1999)
16. Marcus, M., Santorini, B., Marcinkiewicz, M.: Building a large annotated corpus of english: The Penn treebank. Computational Linguistics **19** (1993)
17. Schmid, H.: Efficient parsing of highly ambiguous context-free grammars with bit vectors. In: Proc. of COLING-04. (2004)
18. Black, E., etal.: A procedure for quantitatively comparing the syntactic coverage of English grammars. In: Proc. of DARPA-91. (1991)
19. Chiang, D., Bikel, D.: Recovering latent information in treebanks. In: Proc. of COLING'02. (2002)
20. Ghahramani, Z., Jordan, M.: Factorial Hidden Markov Models. Technical report, MIT (1995)