# Bayesian Model Merging for Unsupervised Constituent Labeling and Grammar Induction

**Gideon Borensztajn**
Institute for Logic, Language and Computation, University of Amsterdam
Plantage Muidergracht 24
1018 TV, Amsterdam, the Netherlands
`gideon@science.uva.nl`

**Willem Zuidema**
Institute for Logic, Language and Computation, University of Amsterdam
Plantage Muidergracht 24
1018 TV, Amsterdam, the Netherlands
`jzuidema@science.uva.nl`

## Abstract

Recent research on unsupervised grammar induction has focused on inducing accurate bracketing of sentences. Here we present an efficient, Bayesian algorithm for the unsupervised induction of syntactic categories from such bracketed text. Our model gives state-of-the-art results on this task, using gold-standard bracketing, outperforming the recent semi-supervised approach of (Haghighi and Klein, 2006), obtaining an $F_1$ of 76.8% (when appropriately relabeled). Our algorithm assigns comparable likelihood to unseen text as the treebank PCFG. Finally, we discuss the metrics used and linguistic relevance of the results.

## 1 Introduction

If the unsupervised induction of grammar is still a key open problem in machine learning, it is not for lack of trying: at least since (Solomonoff, 1964), much talent and effort has been invested in finding algorithms for learning syntactic structure from plain text or semantically or phonetically enriched input data. Almost every combination of reasonable choices for syntactic formalism, search procedure, success criterion and input data has been tried, but little of this work is remembered today. Useful reviews from 3 past decades are (Pinker, 1979; Angluin and Smith, 1983; Sakakibara, 1997). In the last 5 years, new efforts have been made to evaluate such algorithms on manually annotated corpora such as ATIS (Hemphill et al., 1990) and the Penn WSJ corpus (Marcus et al., 1993). An important breakthrough was the CCM algorithm of (Klein and Manning, 2002a), which assigns brackets to sentences in a corpus and was the first to outperform a right-branching base-line onm English. Since then, several other algorithms have been described that also score better than this base-line (Klein and Manning, 2004; Dennis, 2005; Bod, 2006; Seginer, 2007). These models have in common that they do not label constituents with syntactic categories and do not use a generative, PCFG probability model[1].

Although accurate bracketing is important, it is clear that it is only a first, intermediate step. For cognitive plausibility, as well as for most NLP applications, we need at least an account of how constituents are categorized. This task is similar to POS-tagging, and because the latter can be done very accurately with existing techniques, the former is often assumed to be an easy task too. Therefore, it has not received as much attention as it deserves. However, we know of no empirical results that back-up this intuition, and only of one paper (Haghighi and Klein, 2006) that actually evaluates an EM-based unsupervised constituent labeling algorithm (as a baseline for a much better semi-

[1]Instead, these algorithms use (i) a specialized, generative constituent-distituent model (Klein and Manning, 2002a), (ii) a generative dependency model (Klein and Manning, 2004), (iii) a non-generative exemplar-based model (Dennis, 2005), (iv) a generative tree-substitution model (Bod, 2006) and a non-generative dependency-like model (Seginer, 2007).

supervised model), but with disappointing results (see section 7). Even the issue of the evaluation of the categories is by no means trivial, and no standardized measure is agreed upon until now.

In this paper, we develop an unsupervised constituent labeling algorithm that outperforms the EM-based algorithm and has comparable performance to the supervised treebank PCFG on the development set. We base ourselves on the elegant framework proposed by (Stolcke, 1994; Stolcke and Omohundro, 1994), called Bayesian Model Merging (henceforth, BMM). Unlike this work, our algorithm takes bracketed sentences as input; like the full BMM, our algorithm proceeds by merging nonterminal labels to maximize a Bayesian objective function. The initial conditions, merge operation and objective function are described in section 2. In section 3 we give our optimizations that allow for empirical evaluation on large, benchmark corpora, and in sections 4 experimental results and an analysis of the model's strengths and weaknesses. Apart from evaluating the categories against a manually annotated gold standard, we are also interested in a theory-independent measure of performance. In section 4 we present results that compare the likelihood of the parses of the grammar induced by our labeling algorithm to the parses of the treebank grammar on a test set. Finally, in section 5 we describe a version of the algorithm that does complete unsupervised induction, involving both labeling and bracketing, but show that our objective function is not appropriate for this task.

## 2 Search and Objective Function

BMM defines a heuristic, greedy search for an optimal probabilistic context free grammar (PCFG) according to the Bayesian criterion of maximum a posteriori probability. In the present version, a single operator $\text{MERGE}(G, X_1, X_2) \mapsto G'$ defines possible transitions between grammars (Cook et al., 1976): it replaces non-terminal $X_2$ with non-terminal $X_1$ throughout grammar $G$, yielding a new grammar $G'$. MERGE creates generalizations by forming disjunctive groups (categories) of patterns that occur in the same contexts. In the search, all candidate merges are considered, and a single one is selected that most improves the objective function. The process is iterated until a state is reached where no single merge improves the objective function anymore. The search is augmented with a look-ahead procedure, that looks at a sequence of by default 10 subsequent merges.

The algorithm takes as input unlabeled sentences, with bracket information from the treebank (or from a specialized unsupervised bracketing algorithm). The initial rules of the grammar are read off from all productions implicit in the bracketed corpus, where every constituent, except for the start symbol, is given a unique label. The vast number of unique labels thus obtained is reduced to about half its size by merging, in a preprocessing step, labels of constituents which have exactly the same descendants. For example, the annotated sentence *(S (NP-SBJ (NNP Mr.) (NNP Ehrlich)) (VP (MD will) (VP (VB continue) (PP-CLR (IN as) (NP (NP (DT a) (NN director)) (CC and) (NP (DT a) (NN consultant))))))))*, is incorporated in the grammar as follows:

$$\begin{array}{lll|lll} S & \rightarrow X_0\, X_1 & (1) & & & \\ X_0 & \rightarrow NNP\, NNP & (1) & X_3 & \rightarrow IN\, X_4 & (1) \\ X_1 & \rightarrow MD\, X_2 & (1) & X_4 & \rightarrow X_5\, CC\, X_5 & (1) \\ X_2 & \rightarrow VB\, X_3 & (1) & X_5 & \rightarrow DT\, NN & (2) \end{array} \qquad (1)$$

Possible merges are evaluated by the posterior probability of the resulting grammar. The maximum a posteriori (MAP) hypothesis, $M_{MAP}$ is the hypothesis that maximizes the posterior probability. With Bayes Law: $M_{MAP} \equiv argmax_M P(M|X) = argmax_M P(X|M) \cdot P(M)$ where $P(X|M)$ is the likelihood of data $X$ given grammar $M$, and $P(M)$ is the prior probability of the grammar. Maximizing $P(X|M) \cdot P(M)$ is equivalent to minimizing

$$-logP(M) - logP(X|M) \approx GDL + DDL = DL$$

This equation is interpreted in information theory as the total description length (DL): The Grammar Description Length $GDL = -logP(M)$ is the number of bits needed to encode the grammar (rounded to an integer number and assuming an optimal, shared code) and the Data Description Length $DDL = -logP(X|M)$ is code-length needed to describe the data given the model (Solomonoff, 1964). The MAP hypothesis is therefore the grammar with *Minimum Description Length*.

**Structure Prior** Using the description length interpretation allows for an intuitive way to choose the prior probability such that smaller grammars are favored[2]. We adopted the encoding scheme from (Petasis et al., 2004), which divides the grammar into top-productions (the set $R_1$), lexical productions ($R_2$) and other non-lexical productions ($R_3$). Rules from $R_1$ need one symbol less to encode than rules from $R_3$, because their LHS is fixed. $N_r$ is the number of non-terminals in the RHS of a production $r$. Each non-terminal symbol requires $log(\mathcal{N}+1)$ bits to encode, where $\mathcal{N}$ is the number of unique nonterminals, and the 1 is for an end marker. $T$ is the number of terminals ($|R_2| = T$), and 2 further end symbols are needed as separators of the three rule sets. Hence, the grammar description length is given by:

$$\begin{aligned} GDL &= \log(\mathcal{N}+1) \cdot \textstyle\sum_{r \in R_1} (N_r + 1) + (\log(\mathcal{N}+1) + \log(T)) \cdot T \\ &+ \log(\mathcal{N}+1) \cdot \textstyle\sum_{r \in R_3} (N_r + 2) + \log(\mathcal{N}+1) \cdot 2 \end{aligned} \tag{2}$$

**Likelihood** Assuming independence between the sentences, the likelihood of the corpus is the product of the likelihood of the sentences.

$$P(X|M) = \prod_{x \in X} \sum_{der:\ yield(der)=x} P(der|M) \tag{3}$$

The BMM algorithm makes two further approximations in the calculation of the likelihood (Stolcke and Omohundro, 1994). First, it is assumed that most of the probability mass of the sentence is concentrated in the Viterbi parse (the most probable derivation), so that the contribution of all non-Viterbi parses to the sentence probability and hence to the likelihood are ignored. Secondly, it is assumed that the merging operation preserves the Viterbi parse. This means that after a merge operation the Viterbi parse of the sentence is generated by exactly the same sequence of rewrite rules as before, except for the rules affected by the merge. We will come back to the validity of these approximations in section 5.

Using these approximations, we can compute the data likelihood directly from the grammar, if we keep track of the number of times that every rule is used in the entire corpus. This can be seen by rearranging the equation of likelihood, regrouping the rules used in all the samples according to their left hand side (Stolcke, 1994):

$$\begin{aligned} P(X|M) &= \textstyle\prod_{x \in X} \sum_{yield(der)=x} P(der) \approx \prod_{x \in X} P(der(X)_V) \\ &= \textstyle\prod_{x \in X} \prod_{r_i \in der_V} P(r_i)^{C_i} \prod_{A \in V_N} \prod_{r_i : A = lhs(r_i)} P(r_i)^{CC_i} \end{aligned} \tag{4}$$

where $der$ is a derivation, $der_V$ is the Viterbi parse, $r_i \in \mathcal{R}$ is a rewrite rule, $A \in V_N$ a nonterminal, $C_i$ the count of rules occurring within a single derivation and $CC_i$ the count of rules occurring in the Viterbi parses of the entire corpus.

## 3 Forecasting DL-gain

In our search for the grammar that maximizes the objective function, we will need to consider an enormous grammar space. At each time step, the number of alternative grammars reachable with one merge is quadratic in the number of nonterminals. It is therefore computationally intractable to calculate the posterior probability of each candidate grammar. Evaluating the BMM algorithm on realistically sized corpora therefore seemed infeasible (Klein, 2005; Clark, 2001). However, (Petasis et al., 2004) show that the DL-gain of chunks and merges can be efficiently predicted without having to consider a complete alternative grammar for every candidate search operation. Their equations can be adapted for the various choices of objective functions discussed above. The complexity of finding the best chunk is reduced from $O(\mathcal{N}^2)$ to $O(\mathcal{N})$, while the complexity of finding the best merge is reduced from $O(\mathcal{N}^3)$ to $O(\mathcal{N}^2)$ (Petasis et al., 2004).

After the application of a merge, the grammar is made smaller through elimination of duplicate rules: $\Omega_1$ is the set of rules from $R_1$ that are eliminated, and $\Omega_3$ is the set of rules from $R_3$ that are

[2]We are aware that there is much more to be said about the relation between Bayesian Inference and MDL, and that there might be much more linguistically motivated ways to choose priors (see e.g. (Eisner, 2002)). Here we take a pragmatic approach, however, aimed at defining simple priors that nevertheless force the algorithm to generalize beyond the training data.

eliminated. The change of the GDL as a result of the merge is (Petasis et al., 2004):

$$
\begin{aligned}
\Delta GDL_M &= \log\left(\frac{\mathcal{N}}{\mathcal{N}+1}\right) \times \left(\sum_{r\in R_1\cap R_3}(N_r+1)+T+|R_3|+2\right) \\
&- \log(\mathcal{N})\cdot \sum_{r\in\Omega_1\cap\Omega_3}(N_r+1) \qquad (5)
\end{aligned}
$$

As before, the first term expresses the fact that the number of bits needed to encode any single non-terminal is changed due to the decrease in the total number of non-terminals, and this term is independent of the choice of the merge.

For the computation of the gain in data description length (DDL) as a result of merging the non-terminals X and Y, we can best view the merging process as two subprocesses:

$M^1$ the rule sets with LHSs $X$ and $Y$ are joined (receive same LHS), changing the conditional probability of those rules and thus the DDL.
$M^2$ duplicate rules that may occur as a result of the merge are eliminated in the entire grammar.

Although (Petasis et al., 2004) make this observation, their equations for $\Delta DDL$ cannot be used in our model, because they (implicitly) assume rule probabilities are uniformly distributed. That is, their E-GRIDS model assumes CFGs, whereas we work with PCFGs and, like (Stolcke, 1994), assume that rule probabilities are proportional to their frequencies in the Viterbi parses. This requires a modification of the formulas expressing the contribution of the merging operator to DDL.

From eq. 4, we see that the contribution of a non-terminal to the DDL is given by:

$$DDL_X = -\sum_{r:\ X=lhs(r)} F_r \cdot \big(log\left(\frac{F_r}{F_{Tot_X}}\right)\big)$$

where $F_r$ is the frequency of a rule $r$ with LHS $X$, and $F_{Tot_X}$ is the sum of the frequencies of all rules with LHS $X$.

By joining the rules with LHS $X$ and with LHS $Y$ into a single set of rules, the relative frequency of a single rule is changed from $(\frac{F_r}{F_{Tot_X}})$ to $(\frac{F_r}{F_{Tot_{X+Y}}})$. This results in an overall gain in DDL of:

$$\Delta DDL_{M^1} = -\sum_{r:X=lhs(r)}\left(F_r\cdot\log(\frac{F_{Tot_X}}{F_{Tot_{X+Y}}})\right) - \sum_{r:X=lhs(r)}\left(F_r\cdot\log(\frac{F_{Tot_Y}}{F_{Tot_{X+Y}}})\right) \quad (6)$$

The gain in DDL from elimination of duplicate rules is given by:

$$
\begin{aligned}
\Delta DDL_{M^2} &= -\sum_{W\in\theta}\sum_{\omega\in\Omega_W}\Big[\sum_{r\in\omega}F_r\cdot\log(\sum_{k\in\omega}F_k/\sum_{l:lhs(l)=W}F_l) \\
&\quad -\sum_{r\in\omega}\big(F_r\cdot\log(F_r/\sum_{l:lhs(l)=W}F_l)\big)\Big] \qquad (7)
\end{aligned}
$$

where $\theta$ is the set of LHS nonterminals of the duplicate rules resulting from merging $X$ and $Y$. We thus sum over all LHS non-terminals $W$ that have duplicate rules, and over all sets $\omega$ of duplicate rules. (For efficiency, our implementation maintains a datastructure that represents the DL gain of every pair of nonterminals that can be potentially merged, and updates this at every merge.)

## 4 Metrics and experimental evaluation

The BMM algorithm was trained and evaluated on the entire WSJ10 corpus. WSJ10 is the portion of 7422 sentences of length $\leq 10$, after removal of punctuation and traces, extracted from the Penn Treebank Wall Street Journal (WSJ) (Marcus et al., 1993). It has been the prime benchmark used in recent grammar induction research. The POS-sequences are used as input for the induction algorithm, resulting in a vocabulary of 35 POS-tags. We left out sentences consisting of a single word and sentences consisting of a repetition of the same word, to avoid spurious merges. Using the optimizations described above, the run on the 7422 sentences of the WSJ10 corpus lasted approximately 10 hours on a PC with 0.5 Gb memory.

Evaluating the quality of induced syntactic categories is difficult, and no widely agreed upon measure exists. We use two different metrics, related to those used in supervised parsing (labeled precision/recall) and speech recognition (likelihood/perplexity). For the first, the difficulty is that in unsupervised labeling algorithms, categories receive arbitrary internal labels; in order to evaluate them using labeled precision and recall, the induced labels must somehow be mapped onto the treebank labels. We follow (Haghighi and Klein, 2006), who use for their unsupervised baseline model a greedy remapping of the induced labels to the best matching tree bank label. This style of remapping, however, allows for multiple induced labels to map to a single target label. With a fixed number of categories that is no major problem, but with many more induced non-terminals the measure is too optimistic: in the extreme case of a unique induced label for every constituent it would give 100% precision and recall. Since in most of our experiments the number of experimental labels exceeded the number of treebank labels by a large number, we measured labeled recall by defining the remapping the other way round, from the treebank labels to the induced labels. That is, we replace each tree bank label with its best matching induced label, and measure recall with this transformed treebank as gold standard. The F-score is still defined as the harmonic mean of LP and LR: $F_1 = \frac{2*LP*LR}{LP+LR}$.

The motivation for this way of calculating precision and recall is somewhat involved. Consider that syntactic categories are meant to be defining substitutability. Every label $X$ that is used for $N_X$ constituents in the induced trees, thus defines $N_X \cdot (N_X - 1)$ substitutions that are grammatical according to the induced grammar $G_i$. If out of these $N_X$ constituents, $M_{X,Y}$ constituents receive the same label $Y$ in the gold-standard treebank, that means that at least $M_{X,Y} \cdot (M_{X,Y} - 1)$ of the substitutions that are grammatical according to $G_i$ are also grammatical according to the gold-standard grammar $G_g$. Although there might be other categories $Y'$ that permit other substitutions, the portion of the $N_X \times (N_X - 1)$ substitutions that is also permitted by $G_g$ will be dominated by $Z_{max} = \text{argmax}_Z M_{X,Z}$. Hence, $M_{Z_{max}}$ is a lower bound and a good approximation of the square root of the number of $X$-substitutions permitted by $G_g$ ("substitutability precision"). Similar reasoning on the number of treebank substitution permitted by $G_i$ ("substitutability recall") leads to the measures proposed (ignoring for the moment some issues with averaging results).

A drawback of this style of evaluating, is that it is still completely dependent on the manual treebank annotation, which is far from theory-neutral. In fact, all state-of-the-art supervised parsing and language models, can be viewed as redefining the treebank nonterminal labels. Our second style of evaluation avoids this dependency. It is based on splitting the corpus in a train and test set, and measuring the likelihood (or perplexity) of the test set according to the induced grammar. Grammars that overfit the trainset, or generalize too much to unlikely or ungrammatical sentences, will give low likelihood to test set sentences. For these experiments we use the traditional sections 2-21 of WSJ10 as trainset, and section 22 as test set (section 23 is kept for future evaluations).

In table 1(a) we report our relabeled recall and precision scores. For comparison, we also give the baseline results (Haghighi and Klein, 2006) obtained by running the Inside-Outside algorithm on grammar initialized with all binary rules that can be built from the treebank syntactic categories. We also copy their results with a semi-supervised "proto-type" driven induction algorithm run on the same data. Our unsupervised algorithm obtained a (relabeled) F-score of 76.8, which compares favorably to the (labeled) F-score of 71.1 from that paper. Note however, that different definitions of LP and LR are used. In table 1(b) we give the relabeled precision scores of the 7 most frequent categories (after relabeling) in our induced trees. For most frequent categories (leaving out the S category) precision is near or above 90% except for the NP category.

For a second set of experiments, the algorithm was trained on sections 2-21 of WSJ10, and evaluated on section 22. Figure 1(a) gives the sum of the likelihood of all testset sentences, as parsed by the baseline "unlabeled" (the treebank PCFG after all nonterminals have been replaced by an X), the treebank PCFG (labeled TBG) and the grammar that results from our BMM-algorithm (the latter grammar failed to parse 6 sentences up to length 20, and 4 up to length 10. These were excluded from all evaluations.) Figure 1(b) shows that the vast majority of the sentences in the test set receives higher likelihood from the grammar induced by BMM than from either the treebank PCFG or the unlabeled grammar. Note that the grammar that uses a unique nonterminal for every constituent in

(a)

| Model | LP | LR | F |
|---|---|---|---|
| **In-Out** | 47.0 | 57.2 | 51.6 |
| **Proto** | 64.8 | 78.7 | 71.1 |
| **BMM (all)** | 75.1 | 78.5 | 76.8 |

(b)

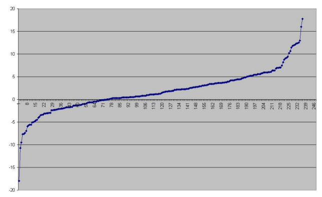
| Label | LP | Freq |
|---|---|---|
| NP | 57.8 | 38.5% |
| VP | 92.0 | 22.3% |
| PP | 89.0 | 8.1% |
| ADVP | 100.0 | 4.0% |
| ADJP | 100.0 | 2.7% |
| QP | 89.3 | 1.6% |
| SBAR | 100.0 | 1.4% |

Table 1: (a) Results of several algorithms on the unsupervised labeling task (using gold standard bracketing). The results with the unsupervised Inside-Outside algorithm and the semi-supervised Prototype-Driven Grammar Induction-algorithm (labeled Proto) are from (Haghighi and Klein, 2006). (b) Relabeled precision scores per category (after mapping to the treebank categories).

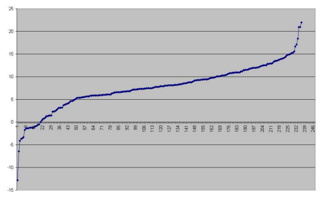
the treebank, such as BMM uses as initial condition, would score even worse: it assigns a likelihood of 0 to all unseen sentences (i.e. -log likelihood of $\infty$ to the testset).

| $G$ | $s \in$ WSJ10 | WSJ20 |
|---|---|---|
| TBG | 7196 | 47266 |
| BMM | 6783 | 46612 |
| unlab | 8592 | 55961 |

(a) $\sum_s -\log P(s|G)$



(b) TBG-BMM



(c) unlabeled-BMM

Figure 1: (a) The summed -log likelihood of sentences from section 22, (b) Log likelihood differences between TBG and BMM on all sentences of WSJ10 (y-axis gives the P(sen—G), x-axis gives rank when sentences are ordered on y-axis value), (c) Log likelihood differences between the unlabeled grammar and BMM.

## 5 Combining Unsupervised Labeling with Unsupervised Bracketing

We have also tried to use the BMM algorithm for the task of simultaneous bracketing and labeling. For this purpose, we add the chunk operator: $\text{CHUNK}(G, X_1, X_2) \mapsto G'$, which takes a sequence of two nonterminals $X_1$ and $X_2$ and creates a unique new nonterminal $Y$ that expands to $X_1X_2$. The merging and chunking operations now proceed in two alternating phases: in the merging phase, the search is iterated until a state is reached where no single merge improves the objective function anymore. In the chunking phase only one best chunk is selected. As in (Stolcke and Omohundro, 1994), the initial rules of the grammar are set to incorporate the complete samples, and the bracketing information from the treebank is discarded.

By adding the chunk operation, we have effectively recovered the full Bayesian Model Merging algorithm of (Stolcke and Omohundro, 1994). Because of our optimizations, we are able to perform the first evaluation of that algorithm on the benchmark test WSJ10. Tables 2 summarize results from experiments with the BMM algorithm on WSJ10 (our best results use the "Poisson prior" with $\mu = 3$; for details see (Stolcke, 1994)). Unfortunately, the completely unsupervised BMM algorithm does not meet up to the standards of previous work on unsupervised bracketing and even to the right branching baseline (R-B).

We investigated a number of possible explanations for the poor performance. First, we considered the possibility that our assumption that Viterbi parses are mostly preserved after the application of chunks and merges (section 2) does not hold. We parsed the entire corpus again with the induced grammar, and compared the resulting parses with the parses assumed in the approximation. The differences were relatively minor when measured with the same metric as used for comparison

(a)

| **WSJ-10** | **UP** | **UR** | **F** |
|---|---|---|---|
| **R-B** | 70.0 | 55.1 | 61.7 |
| **CCM (Klein and Manning, 2002b)** | 64.2 | 81.6 | 71.9 |
| **DMV+CCM (Klein and Manning, 2004)** | 69.3 | 88.0 | 77.6 |
| **U-DOP (Bod, 2006)** | 70.8 | 88.2 | 78.5 |
| **E-GRIDS (Petasis et al., 2004)** | 59.3 | 37.8 | 46.2 |
| **BMM (this paper)** | **57.6** | **43.1** | **49.3** |

(b)

| ( $\times 10^6$) | **DL** | **GDL** | **DDL** |
|---|---|---|---|
| **Init** | .295 | .066 | .226 |
| **Final** | .276 | .041 | .235 |
| | **UP** | **UR** | **F** |
| **Init** | 90.1 | 88.6 | 89.4 |
| **Final** | 64.3 | 74.8 | 69.2 |

Table 2: (a) Results of BMM compared to previous work on the same data; (b) BMM initialized with the treebank grammar

against gold-standard parses, with $UP = UR = 95.9$ on OVIS, and $UP = 95.1$ and $UP = 94.2$ on WSJ10. Hence, our approximations seem to be justified. A second possibility is that the heuristic search procedure used is unable to find the high-accuracy regions of the search space. To test this, we used the treebank PCFG as the initial grammar of the BMM algorithm, which yields comparatively very high UP and UR scores. As table 2(b) shows, this treebank grammar is not an optimum of the objective function used: the BMM algorithm continues for a long time to improve the description length, whilst the F-score against the gold standard parses monotonously decreases.

A qualitative analysis of the merges and chunks in the process further shows a number of problems. First, there are many ungrammatical chunks, which are formed by cutting across constituent boundaries, e.g. $put\ the$, $on\ the$, $and\ a$, $up\ and$. This is explained by the fact that the prime criterion for selecting a chunk is the bigram frequency. Second, there is a tendency for categories to overgeneralize. This effect seems to be self-reinforcing. Merging errors are carried over to the chunking phase and vice versa, causing a snow ball effect. Eventually, most categories cluster together. These experiments indicate that it is not a failure of the approximations or the search algorithm which prevents the algorithm from reaching the optimal grammar, but rather a wrong choice of the objective function. Better choices for the prior probability distribution are a major topic for future research.

## 6 Discussion and Conclusions

In this paper we have studied the problem of labeling constituents in a bracketed corpus. We have argued that this is an important task, complementing the task of unsupervised bracketing on which much progress has been made in recent years (Klein and Manning, 2002b; Bod, 2006). Together, unsupervised bracketing and unsupervised labeling hold the promise of (i) accounting for the unsupervised acquisition of grammar by children, and (ii) relieving research in (multilingual) NLP of its sparseness of annotated data. Our algorithm uses the PCFG formalism and starts out with a grammar that models the training data perfectly, but does not generalize beyond it. It is important to note that the right sequence of merges can take us to any PCFG consistent with the bracketing. Although natural language contains some constructions that are difficult or, in exceptional cases, impossible to model with PCFGs, the formalism is rich enough to encode extremely accurate grammars if (and only if) the traditional linguistic categories are dropped (see e.g. (Petrov et al., 2006))[3]. The key question is thus how to guide the model merging algorithm through the space of possible PCFGs.

[3]For smoothing, (Petrov et al., 2006) use information internal to non-terminal labels, which takes their model outside the class of PCFGs. However, without smoothing they already obtain surprisingly good results.

We have used a Bayesian objective function, in the tradition of (Solomonoff, 1964), to guide this search, and implemented a number of heuristics to perform this search efficiently, using techniques from (Cook et al., 1976), (Stolcke and Omohundro, 1994) and (Petasis et al., 2004). We have found that our BMM-algorithm performs better than state-of-the-art unsupervised labeling algorithms on (re)labeled precision and recall metrics. We have further shown that the likelihood our induced grammars assign to unseen sentences, rivals that of the treebank PCFG (although it presumably performs worse than state-of-the-art supervised language models). These positive results suggest it is possible to devise hybrid models, where specialized bracketers such as (Klein and Manning, 2004) and (Bod, 2006), can be combined with BMM as a specialized unsupervised labeling algorithm.

In section 5, we have demonstrated briefly that, contrary to received wisdom (Klein and Manning, 2005), (Clark, 2001) the full BMM framework of (Stolcke and Omohundro, 1994), which includes bracketing, can be evaluated on large corpora. We think this is an important step in its own right. There is a rich body of research from previous decades on unsupervised grammar induction. We have shown that it can be worthwhile to use an older algorithm, and evaluate it according to current experimental methodology. However, the performance of completely unsupervised BMM on real languages is rather disappointing. The fact that BMM can still optimize the description length when initialized with the treebank grammar indicates that the problem is probably not with the search, but with the applicability of the objective function for natural languages. The distinction BMM makes between prior, likelihood and heuristic search, allows us to now focus our attention on the prior, without having to replace the entire technical apparatus we and others developed.

The finding that the same objective function is appropriate for merging, but not at all for chunking, is interesting in its own right. Because the structure of natural language reflects the learning biases of language learners (Kirby et al., 2007; Zuidema, 2003), this finding might be relevant for theories of language acquisition. Perhaps children use distributional information, such as exploited by our algorithm, to a large extent for discovering syntactic categories, while relying on other information (semantic integrity, phonological phrasing) for identifying constituents in the first place. The finding that induced syntactic categories outperform traditional linguistic categories is in concordance with theories in cognitive linguistics that deny universal status of these categories, and view them as derived from specific linguistic constructions rather than as a-priori given (Croft, 2001).

**Acknowledgements** WZ is funded by the Netherlands Organisation for Scientific Research (EW), project nr. 612.066.405; many thanks to Yoav Seginer, Remko Scha and Rens Bod for their comments.

## References

Angluin, D. and Smith, C. H. (1983). Inductive inference: Theory and methods. *Computing Surveys*, 15(3).

Bod, R. (2006). An all-subtrees approach to unsupervised parsing. *Proceedings ACL-COLING'06*.

Clark, A. (2001). *Unsupervised Language Acquisition: Theory and Practice*. PhD thesis, University of Sussex.

Cook, C., Rosenfeld, A., and Aronson, A. (1976). Grammatical inference by hill climbing. *Informational Sciences (now: Information Sciences)*, 10:59–80.

Croft, W. (2001). *Radical construction grammar: syntactic theory in typological perspective*. Oxford University Press, Oxford.

Dennis, S. (2005). An exemplar-based approach to unsupervised parsing. In Bara, B. G., Barsalou, L., and Bucciarelli, M., editors, *Proceedings of the 27th Conference of the Cognitive Science Society*. Lawrence Erlbaum.

Eisner, J. (2002). Transformational priors over grammars. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 63–70, Philadelphia. Association for Computational Linguistics.

Haghighi, A. and Klein, D. (2006). Prototype-driven grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*.

Hemphill, C., Godfrey, J., and Doddington, G. (1990). The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*. Morgan Kaufman, Hidden Valley.

Kirby, S., Dowman, M., and Griffiths, T. L. (2007). Innateness and culture in the evolution of language. *PNAS*, 104(12):5241–5245.

Klein, D. (2005). *The Unsupervised Learning of Natural Language Structure*. PhD thesis, Stanford University.

Klein, D. and Manning, C. D. (2002a). A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the ACL*.

Klein, D. and Manning, C. D. (2002b). Natural language grammar induction using a constituent-context model. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Klein, D. and Manning, C. D. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42th Annual Meeting of the ACL*.

Klein, D. and Manning, C. D. (2005). Natural language grammar induction with a generative constituent-context model. *Pattern Recognition*, 38.

Marcus, M., Santorini, B., and Marcinkiewicz, M. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

Petasis, G., Paliouras, G., Karkaletsis, V., Halatsis, C., and Spyropoulos, C. (2004). E-grids: Computationally efficient grammatical inference from positive examples. *Grammars*, 7:69–110.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings ACL-COLING'06*, pages 443–440. Association for Computational Linguistics Morristown, NJ, USA.

Pinker, S. (1979). Formal models of language learning. *Cognition*, 7:217–283.

Sakakibara, Y. (1997). Recent advances of grammatical inference. *Theoretical Computer Science*, 185:15–45.

Seginer, Y. (2007). Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391, Prague, Czech Republic. Association for Computational Linguistics.

Solomonoff, R. (1964). A formal theory of inductive inference, part ii. *Information and Control*, 7(2):224–254.

Stolcke, A. (1994). *Bayesian Learning of Probabilistic Language Models*. PhD thesis, Dept. of Electrical Engineering and Computer Science, University of California at Berkeley.

Stolcke, A. and Omohundro, S. M. (1994). Inducing probabilistic grammars by Bayesian model merging. In *Proc. Second International Colloquium on Grammatical Inference and Applications (ICGI'94)*, volume 862 of *Lecture Notes in Computer Science*, pages 106–118, Berlin. Springer-Verlag.

Zuidema, W. (2003). How the poverty of the stimulus solves the poverty of the stimulus. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15 (Proceedings of NIPS'02)*, pages 51–58. MIT Press, Cambridge, MA.