
Bayesian Model Merging for Unsupervised Constituent Labeling

NO AUTHOR INFORMATION

Abstract

We present an efficient, Bayesian algorithm for unsupervised induction of context-free grammars from plain text, that combines features of the models of [1] and [2]. Unlike recent alternatives our model finds both a constituency-structure and a labeling for the constituents, and uses a generative PCFG probability model. Using a description length prior over grammars, we score worse than specialized bracketing algorithms on the WSJ10 benchmark test. In follow-up experiments we find that the poor performance is not a search problem. Rather, the description length prior favors incorrect bracketing, leading to poor performance even if the treebank PCFG is used as initial grammar. Surprisingly, the same model gives state-of-the-art results on the unsupervised labeling task, using gold-standard bracketing, outperforming the recent semi-supervised approach of [3], obtaining an F_1 of 76.8% (when appropriately relabeled).

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 [4], 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 [5, 6, 7].

In the last 5 years, new efforts have been made to evaluate such algorithms on manually annotated corpora such as ATIS [8] and the Penn WSJ corpus [9]. An important breakthrough was the CCM algorithm of [?], which assigns brackets to sentences in a corpus and was the first to outperform a right-branching base-line. Since then, several other algorithms have been described that also score better than this base-line [11, 12, 13]. These models have in common that they do not label constituents with syntactic categories and do not use a generative, PCFG probability model¹.

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. However, we know of no empirical results that back-up this intuition, and only of one paper [3] that actually evaluates an EM-based unsupervised constituent labeling algorithm (as a baseline for a much better semi-supervised model), but with disappointing results (see section 7). Unfortunately, existing full CFG-induction algorithms that combine bracketing and labeling, such as those of [14] and [15] score poorly, even on metrics that only evaluate the bracketing.

¹Instead, these algorithms use (i) a specialized, generative constituent-distituent model [10], (ii) a generative dependency model [11], (iii) a non-generative exemplar-based model [12] and (iv) a generative STSG model [13].

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. We base ourselves on the elegant framework proposed by [16, 1], 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 ?? and 8 experimental results and an analysis of the model’s strengths and weaknesses. Finally, in section 7 we describe a version of the algorithm that works with plain sentences, but show that our objective function is not appropriate for the task of bracketing.

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. A single operator **Merge** defines possible transitions between grammars [17]: it replaces two existing non-terminals X_1 and X_2 with a single new non-terminal Y . Merge creates generalizations by forming disjunctive groups (categories) of patterns that occur in the same contexts.

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:

$$\begin{array}{lcl}
 S & \rightarrow & X_0 X_1 & (1) \\
 X_0 & \rightarrow & NNP NNP & (1) \\
 X_1 & \rightarrow & MD X_2 & (1) \\
 X_2 & \rightarrow & VB X_3 & (1)
 \end{array}
 \left|
 \begin{array}{lcl}
 X_3 & \rightarrow & IN X_4 & (1) \\
 X_4 & \rightarrow & X_5 CC X_5 & (1) \\
 X_5 & \rightarrow & DT NN & (2)
 \end{array}
 \right.
 \quad (1)$$

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: $M_{MAP} \equiv \operatorname{argmax}_M P(M|X) = \operatorname{argmax}_M P(X|M) \cdot P(M)$ 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

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

This equation is interpreted in information theory as the total description length (DL): The Grammar Description Length $GDL = -\log P(M)$ 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 $DDL = -\log P(X|M)$ is code-length needed to describe the data given the model [4]. The MAP hypothesis is therefore the grammar with *Minimum Description Length*.

Prior Using the description length interpretation allows for an intuitive way to choose the prior probability such that smaller grammars are favored². We adopted the encoding scheme from [2], 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

²We 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. Here we take a pragmatic approach, however, aimed at defining simple priors that nevertheless force the algorithm to generalize beyond the training data.

end symbols are needed as separators of the three rule sets. Hence, the grammar description length is given by:

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

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

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

Our algorithm makes two further approximations in the calculation of the likelihood. 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 8.

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 [16]:

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

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 Stolcke's algorithm on realistically sized corpora therefore seems infeasible [20, 21].

However, [2] 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 $O(\mathcal{N}^2)$ [2].

After the application of a merge, the grammar is made smaller through elimination of duplicate rules: Ω_1 is the set of rules from R_1 that are eliminated, and Ω_3 is the set of rules from R_3 that are eliminated. The change of the GDL as a result of the merge is [2]:

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

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 [2] make this observation, their equations for ΔDDL 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 [16], 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:

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

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 $\left(\frac{F_r}{F_{Tot_X}} \right)$ to $\left(\frac{F_r}{F_{Tot_{X+Y}}} \right)$. This results in an overall gain in DDL of:

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

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

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

where θ 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 ω of duplicate rules. Using these optimizations, we can run the BMM algorithm on the 7422 sentences of the WSJ10 corpus in approximately 20 minutes on a PC with 0.5 Gb memory.

4 Evaluation

Evaluating the quality of induced syntactic categories is difficult, and no widely agreed upon measure exists. 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 [3], 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 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 $Z_{max} = \operatorname{argmax}_Z M_{X,Z}$. 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 how to average the results per category).

5 Unsupervised Labeling Experiments

In table 1 we report our relabeled recall and precision scores. For comparison, we also give the baseline results [3] 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.

Model	LP	LR	F
Inside-Outside	47.0	57.2	51.6
Proto	64.8	78.7	71.1
BMM (all)	75.1	78.5	76.8

Table 1: 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 [3]. Note that different definitions of LP and LR are used (see text).

In table 2 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

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 2: Relabeled precision scores per category

6 Adding Unsupervised Bracketing

Chunk concatenates repeating patterns. It takes a sequence of two nonterminals X_1 and X_2 and creates a new nonterminal Y that expands to X_1X_2 .

In the default set-up, the initial rules of the grammar are set to incorporate all samples as follows: for each sentence $a_1a_2\dots a_l$, new nonterminals X_1, X_2, \dots, X_l are created and the following productions:

$$\begin{aligned}
 S &\rightarrow X_1, X_2 \dots X_l & (1) \\
 X_1 &\rightarrow a_1 & (1) \\
 X_2 &\rightarrow a_2 & (1) \\
 &\vdots & \\
 X_l &\rightarrow a_l & (1)
 \end{aligned} \tag{8}$$

Subsequently, the structure search proceeds in two alternating phases, involving merging and chunking operations respectively. In the merging phase, 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. In the chunking phase only one best chunk is selected. The search is augmented with a look-ahead procedure, that looks at a sequence of by default 5 subsequent merges. This is particularly useful in the chunking phase, since chunking does usually not result directly in an improvement. The algorithm stops when neither merges nor chunks improve the objective function any further.

Several more sophisticated choices for the prior can be made. Stolcke experimented with a prior where production lengths of rules are drawn from a Poisson distribution with mean μ . In this case the code length of a production of length N_r is $\log(\mathcal{P}(N_r - 1; \mu)) + N_r \log(\mathcal{N})$ bits (\mathcal{P} is the Poisson distribution). The first term replaces the end marker above. Because all productions in R_3 are binary, only the top-productions are really affected by this prior. $N_r - 1$ is used because the minimum production length is 1. The description length of the grammar GDL is computed like before, which gives, after rearranging:

$$\begin{aligned}
 GDL &= \log(\mathcal{N} + 1) \cdot (|R_3| + T + 2) \\
 &+ \log(\mathcal{N}) \cdot \sum_{r \in R_1 \cap R_3} N_r \\
 &+ \sum_{r \in R_1} \mathcal{P}(N_r - 1, \mu) + \log(T) \cdot T
 \end{aligned} \tag{9}$$

Most of our experiments used the same value ($\mu = 3$) for this prior, for which Stolcke obtained best results on artificial data. Stolcke further decomposes the prior into a structure prior and a parameter prior, $P(M) = P(M_S) \cdot P(\Theta_M | M_S)$, and defines Dirichlet distributions over the parameters. We have implemented such parameter priors, but found that they have only very minimal effect on the experimental results so we will omit further details.

Forecasting DL-gain from a chunk Chunking affects only the GDL. After a chunk which replaces the sequence $X Y$ by Z , a rule containing 4 additional symbols is added to R_3 (e.g. $Z \rightarrow X Y STOP$), and the rules of R_1 contain $BF(X, Y)$ non-terminals less (BF=bigram frequency). The set of non-terminals is increased by 1, so every symbol now requires $\log(\mathcal{N} + 2)$ bits to encode. We can derive the following formula for the change in GDL as a result of the chunk C [2]:

$$\begin{aligned}
 \Delta DL_C &= \Delta GDL_C = \log\left(\frac{\mathcal{N} + 2}{\mathcal{N} + 1}\right) \\
 &\times \left(\sum_{r \in R_1 \cap R_3} (N_r + 1) + T + |R_3| + 2 \right) \\
 &+ \log(\mathcal{N} + 2) \cdot (4 - BF_C)
 \end{aligned} \tag{10}$$

The first term corresponds to the change in the number of bits needed to encode a single non-terminal in the grammar: $\log\left(\frac{\mathcal{N} + 2}{\mathcal{N} + 1}\right)$, and it is constant for all possible chunks at any particular point.

7 Unsupervised Full Induction Experiments

Here we report the first results of Bayesian Model Merging on the benchmark tests OVIS and WSJ10-POSTAGS. OVIS [22] is a specialized, homogeneous corpus consisting of annotated Dutch sentences from a public transport information system. It contains 10040 sentences, and has a vocabulary of 946 words. WSJ10 is the portion of 7422 sentences of length ≤ 10 , after removal of punctuation and traces, extracted from the Penn Treebank Wall Street Journal section (WSJ) [9]. It has been the prime benchmark used in recent grammar induction research. The POS-sequences are used as input for the induction algorithm. This results in a vocabulary of 35 POS-tags.

For the evaluation we adopted the measures used by [23], which differs from the standard PARSEVAL metrics in the following ways: the sentence-level (S) bracket is counted, brackets of span one are ignored, multiplicity of identical brackets is ignored, and averaging is done over all the brackets in the treebank (rather than per sentence). The F-score is defined as the harmonic mean of UP and UR: $F_1 = \frac{2 \cdot UP \cdot UR}{UP + UR}$.

Tables 3 and 4 summarize results from experiments with the BMM algorithm on OVIS and WSJ10, using the Poisson prior (equation 9) with $\mu = 3$ (on OVIS, slightly higher scores can be obtained with different choices of μ). The OVIS results are similar to results obtained by [15], although the differences in evaluation metric do not permit direct comparisons. Promising is that we outperform, albeit by only a small margin, both our own reimplementations of the E-GRIDS algorithm [2] and the right-branching heuristic.

OVIS	UP	UR	F
R-B	68.91	66.30	67.58
E-GRIDS	72.08	65.41	68.58
BMM	71.75	68.17	69.91

Table 3: Results of BMM on OVIS. Reported are micro-averaged unlabeled bracketing scores that include the top-bracket.

For WSJ10 the scores are disappointing, compared to previous work on unsupervised bracketing and even to right branching (R-B). In the next section we report on a series of additional experiments aimed at clarifying the causes of the failure of the algorithm.

WSJ-10	UP	UR	F
CCM	64.2	81.6	71.9
DMV+CCM	69.3	88.0	77.6
U-DOP	70.8	88.2	78.5
R-B	70.0	55.1	61.7
E-GRIDS	59.3	37.8	46.2
BMM	57.6	43.1	49.3

Table 4: Results of BMM compared to previous work on the same data

8 Analysis

A qualitative analysis of the merges and chunks in the process 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 over-generalize. Categories that start off as linguistically relevant, at a later stage become noisy. 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. Higher up in the hierarchy very few syntactically interesting chunks are formed. By the time the algorithm reaches the chunking stage, the categories have already become too noisy. Figure 1 shows UP, UR, F and DL measured at each chunk of one run of BMM on the WSJ10 corpus. As is clear from this graph, the performance of the induced grammar very quickly levels off, while the DL continues to decrease until the last chunk. Qualitatively similar observations are made in other studies of PCFG induction algorithms [20]

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 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. We experimented with different settings of the look-ahead procedure (section 2), which considers a number of merges that might follow currently possible chunks, and measures DL-gain over the whole sequence of chunk and merges. However, we found no significant improvements of the performance. More informative was an experiment where we used the treebank PCFG as the initial grammar of the BMM algorithm, which yields comparatively very high UP and

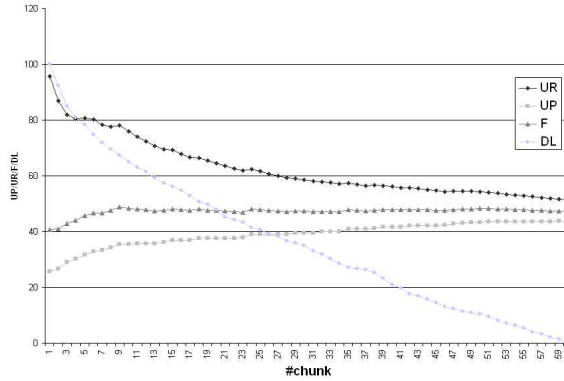


Figure 1: Performance metrics for WSJ10-POSTAGS as a function of chunk number; plotted are unlabeled recall, precision and F-score (see text) and description length (scale omitted).

UR scores. As table 5 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.

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

Table 5: Induction initialized with treebank grammar

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.

9 Conclusions

Our experiments have shown that, contrary to received wisdom [23], [21] the BMM framework of [1] 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.

Surprisingly, BMM performs better than state-of-the-art unsupervised labeling algorithms. This makes it possible to study hybrid models, where specialized bracketers such as [11] and [13] that outperform BMM by a large margin, can be combined with BMM as a specialized unsupervised labeling algorithm.

References

[1] Andreas Stolcke and Stephen M. Omohundro. 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, 1994. Springer-Verlag.
- [2] G. Petasis, G. Paliouras, V. Karkaletsis, C. Halatsis, and C. Spyropoulos. E-grids: Computationally efficient grammatical inference from positive examples. *Grammars*, 7:69–110, 2004.
 - [3] Aria Haghighi and Dan Klein. Prototype-driven grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*. 2006.
 - [4] Ray Solomonoff. A formal theory of inductive inference, part ii. *Information and Control*, 7(2):224–254, 1964.
 - [5] Steven Pinker. Formal models of language learning. *Cognition*, 7:217–283, 1979.
 - [6] Dana Angluin and Carl H. Smith. Inductive inference: Theory and methods. *Computing Surveys*, 15(3), 1983.
 - [7] Yasubumi Sakakibara. Recent advances of grammatical inference. *Theoretical Computer Science*, 185:15–45, 1997.
 - [8] C.T. Hemphill, J.J. Godfrey, and G.R. Doddington. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*. Morgan Kaufman, Hidden Valley, 1990.
 - [9] M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2), 1993.
 - [10] Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the ACL*. 2002.
 - [11] Dan Klein and Christopher D. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42th Annual Meeting of the ACL*. 2004.
 - [12] S. Dennis. An exemplar-based approach to unsupervised parsing. In Bruno G. Bara, Lawrence Barsalou, and Monica Bucciarelli, editors, *Proceedings of the 27th Conference of the Cognitive Science Society*. Lawrence Erlbaum, 2005.
 - [13] Rens Bod. An all-subtrees approach to unsupervised parsing. *Proceedings ACL-COLING'06*, 2006.
 - [14] Pieter Adriaans. *Learning Language from a Categorical Perspective*. PhD thesis, University of Amsterdam, 1992.
 - [15] Menno van Zaanen. *Bootstrapping Structure into Language: Alignment-Based Learning*. PhD thesis, School of Computing, University of Leeds, 2001.
 - [16] A. Stolcke. *Bayesian Learning of Probabilistic Language Models*. PhD thesis, Dept. of Electrical Engineering and Computer Science, University of California at Berkeley, 1994.
 - [17] CM Cook, A Rosenfeld, and AR Aronson. Grammatical inference by hill climbing. *Informational Sciences (now: Information Sciences)*, 10:59–80, 1976.
 - [18] J. Gerard Wolff. Language acquisition, data compression and generalization. *Language & Communication*, 2(1):57–89, 1982.
 - [19] Pat Langley and Sean Stromsten. Learning context-free grammars with a simplicity bias. In *Proceedings of the Eleventh European Conference on Machine Learning*, pages 220–228. Barcelona: Springer-Verlag, 2000.
 - [20] Dan Klein. *The Unsupervised Learning of Natural Language Structure*. PhD thesis, Stanford University, 2005.
 - [21] Alexander Clark. *Unsupervised Language Acquisition: Theory and Practice*. PhD thesis, University of Sussex, 2001.
 - [22] Gert Veldhuijzen van Zanten, Gosse Bouma, Khalil Sima'an, Gertjan van Noord, and Remko Bonnema. Evaluation of the NLP components of the OVIS2 spoken dialogue system. In van Eynde, Schuurman, and Schelkens, editors, *Computational Linguistics in the Netherlands 1998*, pages 213–229. Rodopi, Amsterdam, 1999.
 - [23] Dan Klein and Christopher D. Manning. Natural language grammar induction with a generative constituent-context model. *Pattern Recognition*, 38, 2005.