Unsupervised Learning of Probabilistic Context-Free Grammar using Iterative Biclustering (Extended Version)

Kewei Tu and Vasant Honavar

Department of Computer Science,
Iowa State University, Ames, IA 50011, USA.
{tukw,honavar}@cs.iastate.edu

Abstract. This paper presents PCFG-BCL, an unsupervised algorithm that learns a probabilistic context-free grammar (PCFG) from positive samples. The algorithm acquires rules of an unknown PCFG through iterative biclustering of bigrams in the training corpus. Our analysis shows that this procedure uses a greedy approach to adding rules such that each set of rules that is added to the grammar results in the largest increase in the posterior of the grammar given the training corpus. Results of our experiments on several benchmark datasets show that PCFG-BCL is competitive with existing methods for unsupervised CFG learning.

1 Introduction

Context-free grammars (CFG) constitute an important class of grammars, with a broad range of applications including programming languages, natural language processing, and bioinformatics, among others. A probabilistic context-free grammar (PCFG) is a CFG with probabilities assigned to grammar rules, which can better accommodate the ambiguity and the need for robustness in real-world applications. Hence, the problem of learning a PCFG from data (typically, positive samples generated by the target grammar) is an important problem in grammar induction and machine learning. Several methods for learning (P)CFG from positive data have been proposed. Some rely on different heuristics to iteratively construct an approximation of the unknown CFG [1–5]; others search for a PCFG that has the largest posterior given the training corpus [6–9].

In this paper we propose PCFG-BCL, a new unsupervised algorithm that learns a PCFG from a positive corpus. The proposed algorithm uses (distributional) biclustering to group symbols into non-terminals. This is a more natural and robust alternative to the more widely used substitutability heuristic or distributional clustering, especially in the presence of ambiguity, e.g., when a symbol can be reduced to different nonterminals in different contexts, or when a context can contain symbols of different nonterminals, as illustrated in [1]. PCFG-BCL can be understood within a Bayesian structure search framework. Specifically, it uses a greedy approach to adding rules to a partially constructed grammar, choosing at each step a set of rules that yields the largest possible increase in the posterior of the grammar given the training corpus. The Bayesian framework also supports an ensemble approach to PCFG learning by effectively *combining* multiple candidate grammars. Results of our experiments on several benchmark datasets show that the proposed algorithm is competitive with other methods for learning CFG from positive samples.

The rest of the paper is organized as follows. Section 2 introduces the representation of PCFG used in PCFG-BCL. Section 3 describes the key ideas behind PCFG-BCL. Section 4 presents the complete algorithm and some implementation details. Section 5 presents the results of experiments. Section 6 concludes with a summary and a brief discussion of related work.

2 Grammar Representation

It is well-known that any CFG can be transformed into the Chomsky normal form (CNF), which only has two types of rules: $A \to BC$ or $A \to a$. Because a PCFG is simply a CFG with a probability associated with each rule, it is easy to transform a PCFG into a probabilistic version of CNF.

To simplify the explanation of our algorithm, we make use of the fact that a CNF grammar can be represented in an AND-OR form containing three types of symbols, i.e., AND, OR, and terminals. An AND symbol appears on the left-hand side of exactly one grammar rule, and on the right-hand side of that rule there are exactly two OR symbols. An OR symbol appears on the left-hand side of one or more rules, each of which has only one symbol on the right-hand side, either an AND symbol or a terminal. A multinomial distribution can be assigned to the set of rules of an OR symbol, defining the probability of each rule being chosen. An example is shown below (with rules probabilities in the parentheses).

CNF	**The AND-OR Form**
$S \to a\ (0.4) \mid AB\ (0.6)$	$\mathrm{OR}_S \to a\ (0.4) \mid \mathrm{AND}_{AB}\ (0.6)$
$A \to a\ (1.0)$	$\mathrm{AND}_{AB} \to \mathrm{OR}_A \mathrm{OR}_B$
$B \to b_1\ (0.2) \mid b_2\ (0.5) \mid b_3\ (0.3)$	$\mathrm{OR}_A \to a\ (1.0)$
	$\mathrm{OR}_B \to b_1\ (0.2) \mid b_2\ (0.5) \mid b_3\ (0.3)$

It is easy to show that a CNF grammar in the AND-OR form can be divided into a set of *AND-OR groups* plus the start rules (rules with the start symbol on the left-hand side). Each AND-OR group contains an AND symbol N, two OR symbols A and B such that $N \to AB$, and all the grammar rules that have one of these three symbols on the left-hand side. In the above example, there is one such AND-OR group, i.e., AND_{AB}, OR_A, OR_B and the corresponding rules (the last three lines). Note that there is a bijection between the AND symbols and the groups; but an OR symbol may appear in multiple groups. We may simply make identical copies of such OR symbols to eliminate overlap between groups.

3 Main Ideas

PCFG-BCL is designed to learn a PCFG using its CNF representation in the AND-OR form. Sentences in the training corpus are assumed to be sampled from an unknown PCFG under the i.i.d. (independent and identically distributed) assumption.

Starting from only terminals, PCFG-BCL iteratively adds new symbols and rules to the grammar. At each iteration, it first learns a new AND-OR group by biclustering, as explained in Section 3.1. Once a group is learned, it tries to find rules that attach the newly learned AND symbol to existing OR symbols, as discussed in Section 3.2. This second step is needed because the first step alone is not sufficient for learning such rules. In both steps, once a new set of rules are learned, the corpus is *reduced* using the new rules, so that subsequent learning can be carried out on top of the existing learning result. These two steps are repeated until no further rule can be learned. Then start rules are added to the learned grammar in a postprocessing step (Section 3.3). Since any CNF grammar can be represented in the form of a set of AND-OR groups and a set of start rules, these three steps are capable, in principle, of constructing any CNF grammar.

We will show later that the first two steps of PCFG-BCL outlined above attempt to find rules that yield the greatest increase in the posterior probability of the grammar given the training corpus. Thus, PCFG-BCL performs a local search over the space of grammars using the posterior as the objective function.

3.1 Learning a New AND-OR Group by Biclustering

Intuition. In order to show what it means to learn a new AND-OR group, it is helpful to construct a table T, where each row or column represents a symbol appearing in the corpus, and the cell at row x and column y records the number of times the pair xy appears in the corpus. Because the corpus might have been partially reduced in previous iterations, a row or column in T may represent either a terminal or a nonterminal.

Since we assume the corpus is generated by a CNF grammar, there must be some symbol pairs in the corpus that are generated from AND symbols of the target grammar. Let N be such an AND symbol, and let A, B be the two OR symbols such that $N \to AB$. The set $\{x|A \to x\}$ corresponds to a set of rows in the table T, and the set $\{y|B \to y\}$ corresponds to a set of columns in T. Therefore, the AND-OR group that contains N, A and B is represented by a *bicluster*[10] (i.e., a submatrix) in T, and each pair xy in this bicluster can be reduced to N. See Fig.1 (a), (b) for an example, where the AND-OR group shown in Fig.1(a) corresponds to the bicluster shown in Fig.1(b).

Further, since we assume the target grammar is a PCFG, we have two multinomial distributions defined on A and B respectively that independently determine the symbols generated from A and B. Because the corpus is assumed to be generated by this PCFG, it is easy to prove that the resulting bicluster must be *multiplicatively coherent*[10], i.e., it satisfies the following condition:

$$\frac{a_{ik}}{a_{jk}} = \frac{a_{il}}{a_{jl}} \quad \text{for any two rows } i, j \text{ and two columns } k, l \tag{1}$$

where a_{xy} is the cell value at row x ($x = i, j$) and column y ($y = k, l$).

$\mathrm{AND}_{NP} \to \mathrm{OR}_{Det}\mathrm{OR}_N$

$\mathrm{OR}_{Det} \to \mathrm{the}(0.67) \mid \mathrm{a}(0.33)$

$\mathrm{OR}_N \to \mathrm{circle}(0.2)$

$\quad \mid \mathrm{triangle}(0.3) \mid \mathrm{square}(0.5)$

(a) An AND-OR group (with rule probabilities in the parentheses)

	is	circle	triangle	square	the	⋯
below					8	
above					10	
the		24	36	60		
a		12	18	30		
circle	4					
triangle	6					
⋮						

(b) A part of the table T and the bicluster that represents the AND-OR group. Zero cells are left blank.

	⋯ covers (.)	⋯ touches (.)	⋯ is above (.)	⋯ is below (.)	(.) rolls.	(.) bounces.	⋯
(a, circle)	1	2	1	1	0	0	
(a, triangle)	1	2	1	3	2	1	
(a, square)	3	4	2	4	4	1	⋯
(the, circle)	2	3	1	3	2	1	
(the, triangle)	3	5	2	5	4	2	
(the, square)	5	8	4	8	7	3	

(c) A part of the expression-context matrix of the bicluster

Fig. 1. Example: a bicluster and its expression-context matrix

Given a bicluster in T, we can construct an *expression-context matrix*, in which the rows represent the set of symbol pairs (expressions) in the bicluster, the columns represent all the contexts in which these symbol pairs appear, and the value in each cell denotes the number of times the corresponding expression-context combination appears in the corpus (see Fig.1(c) for an example). Because the target grammar is context-free, if a bicluster represents an AND-OR group of the target grammar, then the choice of the symbol pair is independent of its context and thus the resulting expression-context matrix should also be multiplicatively coherent, i.e., it must satisfy Eq.1.

The preceding discussion suggests an intuitive approach to learning a new AND-OR group: first find a bicluster of T that is multiplicatively coherent and has a multiplicatively coherent expression-context matrix, and then construct an AND-OR group from it. The probabilities associated with the grammar rules can be estimated from the statistics of the bicluster. For example, if we find that the bicluster shown in Fig.1(b) and its expression-context matrix shown in Fig.1(c) are both multiplicatively coherent, we can learn an AND-OR group as shown in Fig.1(a).

Probabilistic Analysis. We now present an analysis of the intuitive idea outlined above within a probabilistic framework. Consider a trivial initial grammar where the start symbol directly generates each sentence of the corpus with equal probability. We can calculate how the likelihood of the corpus given the grammar is changed by extracting a bicluster and learning a new AND-OR group as described above.

Suppose we extract a bicluster BC and add to the grammar an AND-OR group with an AND symbol N and two OR symbols A and B. Suppose there is a sentence d containing a symbol pair xy that is in BC. First, since xy is reduced to N after this learning process, the likelihood of d is reduced by a factor of $P(N \to xy|N) = P(A \to x|A) \times P(B \to y|B)$. Second, the reduction may make some other sentences in the corpus become identical to d, resulting in a corresponding increase in the likelihood. Suppose the sentence d is represented by row p and column q in the expression-context matrix of BC, then this second factor is exactly the ratio of the sum of column q to the value of cell pq, because before the reduction only those sentences represented by cell pq are equivalent to d, and after the reduction the sentences in the entire column become equivalent (the same context plus the same expression N).

Let $LG(BC)$ be the likelihood gain resulting from extraction of BC; let G_k and G_{k+1} be the grammars before and after extraction of BC, D be the training corpus; in the bicluster BC, let A denote the set of rows, B the set of columns, r_x the sum of entries in row x, c_y the sum of entries in column y, s the sum over all the entries in BC, and a_{xy} the value of cell xy; in the expression-context matrix of BC, let EC-row denote the set of rows, EC-col the set of columns, r'_p the sum of entries in row p, c'_q the sum of entries in column q, s' the sum of all the entries in the matrix, and $EC(p, q)$ or a'_{pq} the value of cell pq. With a little abuse of notation we denote the context of a symbol pair xy in a sentence d by $d-$"xy". We can now calculate the likelihood gain as follows:

$$\begin{aligned}
LG(BC) &= \frac{P(D|G_{k+1})}{P(D|G_k)} = \prod_{d \in D} \frac{P(d|G_{k+1})}{P(d|G_k)} \\
&= \prod_{x \in A,\ y \in B,\ xy \text{ appears in } d \in D} P(x|A)P(y|B) \frac{\sum_{p \in \text{EC-row}} EC(p, d - \text{``xy''})}{EC(\text{``xy''}, d - \text{``xy''})} \\
&= \prod_{x \in A} P(x|A)^{r_x} \prod_{y \in B} P(y|B)^{c_y} \frac{\prod_{q \in \text{EC-col}} {c'_q}^{c'_q}}{\prod_{\substack{p \in \text{EC-row} \\ q \in \text{EC-col}}} {a'_{pq}}^{a'_{pq}}}
\end{aligned}$$

It can be shown that, the likelihood gain is maximized by setting:

$$P(x|A) = \frac{r_x}{s} \qquad\qquad P(y|B) = \frac{c_y}{s}$$

Substituting this into the likelihood gain formula, we get

$$\begin{aligned}
\max_{P_r} LG(BC) &= \prod_{x \in A} \left(\frac{r_x}{s}\right)^{r_x} \prod_{y \in B} \left(\frac{c_y}{s}\right)^{c_y} \frac{\prod_{q \in \text{EC-col}} {c'_q}^{c'_q}}{\prod_{\substack{p \in \text{EC-row} \\ q \in \text{EC-col}}} {a'_{pq}}^{a'_{pq}}} \\
&= \frac{\prod_{x \in A} {r_x}^{r_x} \prod_{y \in B} {c_y}^{c_y}}{s^{2s}} \times \frac{\prod_{q \in \text{EC-col}} {c'_q}^{c'_q}}{\prod_{\substack{p \in \text{EC-row} \\ q \in \text{EC-col}}} {a'_{pq}}^{a'_{pq}}}
\end{aligned}$$

where P_r represents the set of grammar rule probabilities. Notice that $s = s'$ and $a_{xy} = r'_p$ (where row p of the expression-context matrix represents the symbol pair xy). Thus we have

$$\max_{P_r} LG(BC) = \frac{\prod_{x\in A} {r_x}^{r_x} \prod_{y\in B} {c_y}^{c_y}}{s^s \prod_{\substack{x\in A\\ y\in B}} {a_{xy}}^{a_{xy}}} \times \frac{\prod_{p\in \text{EC-row}} {r'_p}^{r'_p} \prod_{q\in \text{EC-col}} {c'_q}^{c'_q}}{{s'}^{s'} \prod_{\substack{p\in \text{EC-row}\\ q\in \text{EC-col}}} {a'_{pq}}^{a'_{pq}}}$$

The two factors in the righthand side are of the same form, one for the bicluster and one for the expression-context matrix. This form of formula actually measures the multiplicative coherence of the underlying matrix (in a slightly different way from Eq.18 of [10]), which is maximized when the matrix is perfectly coherent. Therefore, we see that when extracting a bicluster (with the new grammar rule probabilities set to the optimal values), the likelihood gain is the product of the multiplicative coherence of the bicluster and its expression-context matrix, and that the maximal gain in likelihood is obtained when both the bicluster and its expression-context matrix are perfectly multiplicatively coherent. This validates the intuitive approach in the previous subsection. More derivation details can be found in the appendix.

It must be noted however, in learning from data, simply maximizing the likelihood can result in a learned model that overfits the training data and hence generalizes poorly on data unseen during training. In our setting, maximizing the likelihood is equivalent to finding the most coherent biclusters. This can result in a proliferation of small biclusters and hence grammar rules that encode highly specific patterns appearing in the training corpus. Hence learning algorithms typically have to trade off the complexity of the model against the quality of fit on the training data. We achieve this by choosing the prior $P(G) = 2^{-DL(G)}$ over the set of candidate grammars, where $DL(G)$ is the description length of the grammar G. This prior penalizes more complex grammars, as complex grammars are more likely to overfit the training corpus.

Formally, the logarithm of the gain in posterior as a result of extracting an AND-OR group from a bicluster and updating the grammar from G_k to G_{k+1} (assuming the probabilities associated with the grammar rules are set to their optimal values) is given by:

$$\begin{aligned}
&\max_{P_r} LPG(BC) = \max_{P_r} \log \frac{P(G_{k+1}|D)}{P(G_k|D)} \\
&= \left(\sum_{x\in A} r_x \log r_x + \sum_{y\in B} c_y \log c_y - s \log s - \sum_{x\in A, y\in B} a_{xy} \log a_{xy} \right) \\
&\quad + \left(\sum_{p\in \text{EC-row}} r'_p \log r'_p + \sum_{q\in \text{EC-col}} c'_q \log c'_q - s' \log s' - \sum_{\substack{p\in \text{EC-row}\\ q\in \text{EC-col}}} a'_{pq} \log a'_{pq} \right) \\
&\quad + \alpha \left(4 \sum_{x\in A, y\in B} a_{xy} - 2|A| - 2|B| - 8 \right) \qquad (2)
\end{aligned}$$

where $LPG(BC)$ denotes the logarithmic posterior gain resulting from extraction of the bicluster BC; α is a parameter in the prior that specifies how much the prior favors compact grammars, and hence it controls the tradeoff between the complexity of the learned grammar and the quality of fit on the training corpus. Note that the first two terms in this formula correspond to the gain in log likelihood (as shown earlier). The third term is the logarithmic prior gain, biasing the algorithm to favor large biclusters and hence compact grammars (see the appendix for details).

3.2 Attaching a New AND Symbol under Existing OR Symbols

Intuition. For a new AND symbol N learned in the first step, there may exist one or more OR symbols in the current partially learned grammar, such that for each of them (denoted by O), there is a rule $O \to N$ in the target grammar. Such rules cannot be acquired by extracting biclusters as described above: When O is introduced into the grammar, N simply does not exist in the table T, and when N is introduced, it only appears in a rule of the form $N \to AB$. Hence, we need a strategy for discovering such OR symbols and adding the corresponding rules to the grammar. Note that, if there are recursive rules in the grammar, they are learned in this step. This is because the first step establishes a partial order among the symbols, and only by this step can we connect nonterminals to form cycles and thus introduce recursions into the grammar.

Consider an OR symbol O that was introduced into the grammar as part of an AND-OR group obtained by extracting a bicluster BC. Let M be the AND symbol and P the other OR symbol in the group, such that $M \to OP$. So O corresponds to the set of rows and P corresponds to the set of columns of BC.

If $O \to N$, and if we add to BC a new row for N, where each cell records the number of appearances of Nx (for all x s.t. $P \to x$) in the corpus, then the expanded bicluster should be multiplicatively coherent, for the same reason that BC was multiplicatively coherent. The new row N in BC results in a set of new rows in the expression-context matrix. This expanded expression-context matrix should be multiplicatively coherent for the same reason that the expression-context matrix of BC was multiplicatively coherent. The situation is similar when we have $M \to PO$ instead of $M \to OP$ (thus a new *column* is added to BC when adding the rule $O \to N$). An example is shown in Fig.2.

Thus, if we can find an OR symbol O such that the expanded bicluster and the corresponding expanded expression-context matrix are both multiplicatively coherent, we should add the rule $O \to N$ to the grammar.

Probabilistic Analysis. The effect of attaching a new AND symbol under existing OR symbols can be understood within a probabilistic framework. Let $\widetilde{BC}$ be a *derived* bicluster, which has the same rows and columns as BC, but the values in its cells correspond to the expected numbers of appearances of the symbol pairs when applying the current grammar to expand the current partially reduced corpus. $\widetilde{BC}$ can be constructed by traversing all the AND symbols that

AND → $OR_1 OR_2$
OR_1 → big (0.6) | old (0.4)
OR_2 → dog (0.6) | cat (0.4)

New rule: OR_2 → AND

(a) An existing AND-OR group and a proposed new rule

	dog	cat	AND
big	27	18	15
old	18	12	10

(b) The bicluster and its expansion (a new column)

	the (.) slept.	the big (.) slept.	the old (.) slept.	the old big (.) slept.	⋯ heard the (.)	⋯ heard the old (.)	⋯
(old, dog)	6	1	1	0	3	1	⋯
(big, dog)	9	2	1	1	4	1	
(old, cat)	4	1	0	0	2	1	
(big, cat)	6	1	1	0	4	1	
(old, AND)	3	1	0	0	2	1	⋯
(big, AND)	5	1	1	0	2	1	

(c) The expression-context matrix and its expansion

Fig. 2. An example of adding a new rule that attaches a new AND under an existing OR. Here the new AND is attached under one of its own OR symbols, forming a self-recursion.

M can be directly or indirectly reduced to in the current grammar. $\widetilde{BC}$ is close to BC if for all the AND symbols involved in the construction, their corresponding biclusters and expression-context matrices are approximately multiplicatively coherent, a condition that is ensured in our algorithm. Let $\widetilde{BC}'$ be the expanded derived bicluster that contains both $\widetilde{BC}$ and the new row or column for N. It can be shown that the likelihood gain of adding $O \to N$ is approximately the likelihood gain of extracting $\widetilde{BC}'$, which, as shown in Section 3.1, is equal to the product of the multiplicative coherence of $\widetilde{BC}'$ and its expression-context matrix (when the optimal new rule probabilities are assigned that maximize the likelihood gain). Thus it validates the intuitive approach in the previous subsection. See the appendix for details.

As before, we need to incorporate the effect of the prior into the above analysis. So we search for existing OR symbols that result in maximal posterior gains exceeding a user-specified threshold. The maximal posterior gain is approximated by the following formula.

$$\max_{P_r} \log \frac{P(G_{k+1}|D)}{P(G_k|D)} \approx \max_{P_r} LPG(\widetilde{BC}') - \max_{P_r} LPG(\widetilde{BC}) \tag{3}$$

where P_r is the set of new grammar rule probabilities, G_k and G_{k+1} is the grammar before and after adding the new rule, D is the training corpus, $LPG()$ is defined in Eq.2. Please see the appendix for the details.

Algorithm 1 PCFG-BCL: PCFG Learning by Iterative Biclustering

Input: a corpus C
Output: a CNF grammar in the AND-OR form
1: create an empty grammar G
2: create a table T of the number of appearances of each symbol pair in C
3: **repeat**
4: G, C, T, $N \Leftarrow$ LearningByBiclustering(G, C, T)
5: G, C, $T \Leftarrow$ Attaching(N, G, C, T)
6: **until** no further rule can be learned
7: $G \Leftarrow$ Postprocessing(G, C)
8: **return** G

3.3 Postprocessing

The two steps described above are repeated until no further rule can be learned. Since we reduce the corpus after each step, in an ideal scenario, upon termination of this process the corpus is fully reduced, i.e., each sentence is represented by a single symbol, either an AND symbol or a terminal. However, in practice there may still exist sentences in the corpus containing more than one symbol, either because we have applied the wrong grammar rules to reduce them, or because we have failed to learn the correct rules that are needed to reduce them.

At this stage, the learned grammar is almost complete, and we only need to add the start symbol S (which is an OR symbol) and start rules. We traverse the whole corpus: In the case of a fully reduced sentence that is reduced to a symbol x, we add $S \to x$ to the grammar if such a rule is not already in the grammar (the probability associated with the rule can be estimated by the fraction of sentences in the corpus that are reduced to x). In the case of a sentence that is not fully reduced, we can re-parse it using the learned grammar and attempt to fully reduce it, or we can simply discard it as if it was the result of noise in the training corpus.

4 Algorithm and Implementation

The complete algorithm is presented in Algorithm 1, and the three steps are shown in Algorithm 2 to 4 respectively. Algorithm 2 describes the "learning by biclustering" step (Section 3.1). Algorithm 3 describes the "attaching" step (Section 3.2), where we use a greedy solution, i.e., whenever we find a good enough OR symbol, we learn the corresponding new rule. In both Algorithm 2 and 3, a *valid* bicluster refers to a bicluster where the multiplicative coherence of the bicluster and that of its expression-context matrix both exceed a threshold δ. This corresponds to the heuristic discussed in the "intuition" subsections in Section 3, and it is used here as an additional constraint in the posterior-guided search. Algorithm 4 describes the postprocessing step (Section 3.3), wherein to keep things simple, sentences not fully reduced are discarded.

Algorithm 2 LearningByBiclustering(G, C, T)

Input: the grammar G, the corpus C, the table T
Output: the updated G, C, T; the new AND symbol N

find the valid bicluster Bc in T that leads to the maximal posterior gain (Eq.2)
create an AND symbol N and two OR symbols A, B
for all row x of Bc **do**
    add $A \rightarrow x$ to G, with the row sum as the rule weight
for all column y of Bc **do**
    add $B \rightarrow y$ to G, with the column sum as the rule weight
add $N \rightarrow AB$ to G
in C, reduce all the appearances of all the symbol pairs in Bc to N
update T according to the reduction
return G, C, T, N

Algorithm 3 Attaching(N, G, C, T)

Input: an AND symbol N, the grammar G, the corpus C, the table T
Output: the updated G, C, T

for each OR symbol O in G **do**
    if O leads to a valid expanded bicluster as well as a posterior gain (Eq.3) larger than a threshold **then**
        add $O \rightarrow N$ to G
        maximally reduce all the related sentences in C
        update T according to the reduction
return G, C, T

4.1 Implementation Issues

In the "learning by biclustering" step we need to find the bicluster in T that leads to the maximal posterior gain. However, finding the optimal bicluster is computationally intractable [10]. In our current implementation, we use stochastic hill-climbing to find only a fixed number of biclusters, from which the one with the highest posterior gain is chosen. This method is not guaranteed to find the optimal bicluster when there are more biclusters in the table than the fixed number of biclusters considered. In practice, however, we find that if there are many biclusters, often it is the case that several of them are more or less equally optimal and our implementation is very likely to find one of them.

Algorithm 4 Postprocessing(G, C)

Input: the grammar G, the corpus C
Output: the updated G

create an OR symbol S
for each sentence s in C **do**
    if s is fully reduced to a single symbol x **then**
        add $S \rightarrow x$ to G, or if the rule already exists, increase its weight by 1
return G

Constructing the expression-context matrix becomes time-consuming when the average context length is long. Moreover, when the training corpus is not large enough, long contexts often result in rather sparse expression-context matrices. Hence, in our implementation we only check context of a fixed size (by default, only the immediate left and immediate right neighbors). It can be shown that this choice leads to a matrix whose coherence is no lower than that of the true expression-context matrix, and hence may overestimate the posterior gain.

4.2 Grammar Selection and Averaging

Because we use stochastic hill-climbing with random start points to do biclustering, our current implementation can produce different grammars in different runs. Since we calculate the posterior gain in each step of the algorithm, for each learned grammar an overall posterior gain can be obtained, which is proportional to the actual posterior. We can use the posterior gain to evaluate different grammars and perform model selection or model averaging, which usually leads to better performance than using a single grammar.

To perform model selection, we run the algorithm multiple times and return the grammar that has the largest posterior gain. To perform model averaging, we run the algorithm multiple times and obtain a set of learned grammars. Given a sentence to be parsed, in the spirit of Bayesian model averaging, we parse the sentence using each of the grammars and use a weighted vote to accept or reject it, where the weight of each grammar is its posterior gain. To generate a new sentence, we select a grammar in the set with the probability proportional to its weight, and generate a sentence using that grammar; then we parse the sentence as described above, and output it if it's accepted, or start over if it is rejected.

5 Experiments

A set of PCFGs obtained from available artificial, English-like CFGs were used in our evaluation, as listed in the table below. The CFGs were converted into CNF with uniform probabilities assigned to the grammar rules. Training corpora were then generated from the resulting grammars. We compared PCFG-BCL with EMILE [1] and ADIOS [5]. Both EMILE and ADIOS produce a CFG from a training corpus, so we again assigned uniform distributions to the rules of the learned CFG in order to evaluate them.

Grammar Name	Size (in CNF)	Recursion	Source
Baseline	12 Terminals, 9 Nonterminals, 17 Rules	No	Boogie[11]
Num-agr	19 Terminals, 15 Nonterminals, 30 Rules	No	Boogie[11]
Langley1	9 Terminals, 9 Nonterminals, 18 Rules	Yes	Boogie[11]
Langley2	8 Terminals, 9 Nonterminals, 14 Rules	Yes	Boogie[11]
Emile2k	29 Terminals, 15 Nonterminals, 42 Rules	Yes	EMILE[1]
TA1	47 Terminals, 66 Nonterminals, 113 Rules	Yes	ADIOS[5]

We evaluated our algorithm by comparing the learned grammar with the target grammar on the basis of *weak generative capacity*. That is, we compare

Grammar Name	PCFG-BCL			EMILE			ADIOS		
	P	R	F	P	R	F	P	R	F
Baseline (100)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	99 (2)	99 (1)
Num-agr (100)	100 (0)	100 (0)	100 (0)	50 (4)	100 (0)	67 (3)	100 (0)	92 (6)	96 (3)
Langley1 (100)	100 (0)	100 (0)	100 (0)	100 (0)	99 (1)	99 (1)	99 (3)	94 (4)	96 (2)
Langley2 (100)	98 (2)	100 (0)	99 (1)	96 (3)	39 (7)	55 (7)	76 (21)	78 (14)	75 (14)
Emile2k (200)	85 (3)	90 (2)	87 (2)	75 (12)	68 (4)	71 (6)	80 (0)	65 (4)	71 (3)
Emile2k (1000)	100 (0)	100 (0)	100 (0)	76 (7)	85 (8)	80 (6)	75 (3)	98 (3)	85 (3)
TA1 (200)	82 (7)	73 (5)	77 (5)	77 (3)	14 (3)	23 (4)	77 (24)	55 (12)	62 (14)
TA1 (2000)	95 (6)	100 (1)	97 (3)	98 (5)	48 (4)	64 (4)	50 (22)	92 (4)	62 (17)

Table 1. Experimental results. The training corpus sizes are indicated in the parentheses after the grammar names. P=Precision, R=Recall, F=F-score. The numbers in the table denote the performance estimates averaged over 50 trials, with the standard deviations in parentheses.

the language of the learned grammar with that of the target grammar in terms of *precision* (the percentage of sentences generated by the learned grammar that are accepted by the target grammar), *recall* (the percentage of sentences generated by the target grammar that are accepted by the learned grammar), and *F-score* (the harmonic mean of precision and recall). To estimate precision and recall, 200 sentences were generated using either the learned grammar or the target grammar (as the case may be), and then parsed by the other grammar.

To ensure a fair comparison, we tuned the parameters of PCFG-BCL, EMILE and ADIOS on a separate dataset before running the evaluation experiments. Table 1 shows the experimental results. Each table cell shows the mean and standard deviation of performance estimates from 50 independent runs. In each run, each algorithm produced a single grammar as the output.

The results summarized in Table 1 show that PCFG-BCL outperformed both EMILE and ADIOS, on each of the test grammars, and by substantial margins on several of them. Moreover, in a majority of the tests, the standard deviations of the performance estimates of PCFG-BCL were lower than those of EMILE and ADIOS, suggesting that PCFG-BCL is more stable than the other two methods. It should be noted however, that neither EMILE nor ADIOS assume the training corpus to be generated from a PCFG, and thus they do not make full use of the distributional information in the training corpus. This might explain in part the superior performance of PCFG-BCL relative to EMILE and ADIOS.

We also examined the effect of grammar selection and grammar averaging (see Section 4.2), on the four datasets where PCFG-BCL did not achieve a perfect F-score on its own. In each case, we ran the algorithm for 10 times and then used the resulting grammars to perform grammar selection or grammar averaging as described in Section 4.2. The results (data not shown) show that grammar selection improved the F-score by 1.5% on average, and the largest increase of 4.4% was obtained on the TA1-200 data; grammar averaging improved the F-score by 3.2% on average, and the largest increase of 9.3% was obtained also on the TA1-200 data. In addition, both grammar selection and averaging reduced the standard deviations of the performance estimates.

6 Summary and Discussion

6.1 Related Work

Several algorithms for unsupervised learning of CFG from only positive samples are available in the literature. EMILE [1] uses a simpler form of biclustering to create new nonterminals. It performs biclustering on an initial table constructed from the unreduced corpus, finding rules with only terminals on the right-hand side; and then it turns to the substitutability heuristic to find high-level rules. In contrast, PCFG-BCL performs iterative biclustering that finds both kinds of rules. ABL [2] employs the substitutability heuristic to group possible constituents to nonterminals. Clark's algorithm [4] uses the "substitution-graph" heuristic or distributional clustering [3] to induce new nonterminals and rules. These techniques could be less robust than the biclustering method, especially in the presence of ambiguity as discussed in Section 1 and also in [1]. Both ABL and Clark's method rely on some heuristic criterion to filter non-constituents, whereas PCFG-BCL automatically identifies constituents as a byproduct of learning new rules from biclusters that maximize the posterior gain. ADIOS [5] uses a probabilistic criterion to learn "patterns" (AND symbols) and the substitutability heuristic to learn "equivalence classes" (OR symbols). In comparison, our algorithm learns the two kinds of symbols simultaneously in a more unified manner.

The inside-outside algorithm [12, 13], one of the earliest algorithms for learning PCFG, assumes a fixed, usually fully connected grammar structure and tries to maximize the likelihood, making it very likely to overfit the training corpus. Subsequent work has adopted the Bayesian framework to maximize the posterior of the learned grammar given the corpus [6, 7], and has incorporated grammar structure search [6, 8]. Our choice of prior over the set of candidate grammars is inspired by [6]. However, compared with the approach used in [6], PCFG-BCL adds more grammar rules at each step without sacrificing completeness (the ability to find any CFG); and the posterior re-estimation in PCFG-BCL is more straightforward and efficient (by using Eq.2 and 3). An interesting recent proposal within the Bayesian framework [9] involves maximizing the posterior using a non-parametric model. Although there is no structure search, the prior used tends to concentrate the probability mass on a small number of rules, thereby biasing the learning in favor of compact grammars.

Some unsupervised methods [14, 15] for learning grammatical structures other than CFG with the goal of parsing natural language sentences also employ some techniques similar to those used in CFG learning.

6.2 Summary and Future Work

We have presented PCFG-BCL, an unsupervised algorithm that learns a probabilistic context-free grammar (PCFG) from positive samples. The algorithm acquires rules of an unknown PCFG through iterative biclustering of bigrams in the training corpus. Results of our experiments on several benchmark datasets show that PCFG-BCL is competitive with the state of the art methods for learning CFG from positive samples. Work in progress is aimed at improving PCFG-BCL e.g., by exploring alternative strategies for optimizing the objective function, and more systematic empirical evaluation of PCFG-BCL on real-world applications (e.g., induction of grammars from natural language corpora) with respect to both weak and strong generative capacity.

Appendix

A Probabilistic Formalization

In this section, we formalize how the learning process changes the posterior probability of the learned grammar given the training corpus.

The prior is defined as follows.

$$P(G) = 2^{-DL(G)}$$

where $DL(G)$ is the description length of the grammar G. In our algorithm we simply assume the same bit length for any symbol and use the length of the direct representation of the grammar as the description length, but other coding methods can also be used. This prior assigns higher probabilities to smaller grammars (the Occam's Razor principle). Since large, complex grammars are more likely to overfit the training corpus, we use this prior to prevent overfitting. This prior was also used in some previous Bayesian grammar learning algorithms [6].

To start with, we define a trivial initial grammar where the start symbol directly generates all the sentences in the training corpus. For each sentence $s_i =< w_1, w_2, \dots, w_n >$ in the training corpus, where each w_j $(1 \leq j \leq n)$ is a terminal, the initial grammar contains the following set of grammar rules.

$$\begin{array}{l}
S \to w_1 S_{i1} \\
S_{i1} \to w_2 S_{i2} \\
S_{i2} \to w_3 S_{i3} \\
\cdots \\
S_{i(n-2)} \to w_{n-1} S_{i(n-1)} \\
S_{i(n-1)} \to w_n
\end{array}$$

where S is the start symbol and each S_{ij} $(1 \leq j \leq n)$ is a nonterminal.

Starting from this initial grammar, our algorithm can be seen as gradually modifying it with the two steps described in the main text (the learning by biclustering step in Section 3.1 and the attaching step in Section 3.2), and we can formulate how such modifications change the posterior.

Notice that the formulation may be different if we use a different initial grammar (e.g., a CNF one), but as far as the initial grammar generates exactly the set of sentences in the training corpus, the difference should be limited to some constants in the formula and the conclusions should remain the same.

A.1 Learning a New AND-OR Group by Biclustering

In this section we formalize how the learning by biclustering step (Section 3.1 in the main text) changes the posterior. Suppose we extract a bicluster BC, create an AND symbol N and two OR symbols A, B, and add a set of grammar rules to the grammar:

$N \to AB$
$A \to x$ for each row x, with the rule probability assigned
$B \to y$ for each column y, with the rule probability assigned

We also reduce all the appearances of all the symbol pairs in BC to N in the corpus, and accordingly, we modify the grammar so that it generates these new "sentences" instead of the old ones. Specifically, for each appearance of each symbol pair xy in BC, in the original grammar there are two rules

$$S_{ij} \to xS_{i(j+1)}$$
$$S_{i(j+1)} \to yS_{i(j+2)}$$

which are now combined into

$$S_{ij} \to NS_{i(j+2)}$$

First, let's look at how the likelihood is changed. For each sentence that's involved in the reduction, its likelihood is changed by two factors. First, the original derivation that generates xy now generates N instead, and then N generates xy with the probability $P(A \to x|A) \times P(B \to y|B)$; so the likelihood of this sentence is reduced by a factor equal to this probability. Second, the reduction may make some other sentences in the corpus become the same as this sentence, so the likelihood is increased by a factor equal to how many times the number of such equivalent sentences increases. Suppose this sentence is represented by row p and column q in the expression-context matrix, then this second factor is exactly the ratio of the sum of column q to the value of cell (p,q), because before the reduction only those sentences represented by that cell are equivalent, and after the reduction the sentences in the whole column become equivalent (the same context plus the same expression N). To sum up, we can formalize the likelihood gain resulted from the grammar modification as follows.

Denote the likelihood gain of extracting BC by $LG(BC)$. Let D be the set of sentences in the training corpus, and let G_k and G_{k+1} be the grammar before and after extracting the bicluster. By abuse of notation we denote the set of rows of BC by A, and the set of columns by B, and denote the context of a symbol pair xy in a sentence d by $d-$"xy". For the bicluster BC, denote the sum of row x by r_x, the sum of column y by c_y. For the expression-context matrix, denote its value at row i and column j by $EC(i,j)$, its set of rows by EC-row, its set of columns by EC-col, and the sum of column q by c'_q.

$$LG(BC) = \frac{P(D|G_{k+1})}{P(D|G_k)} = \prod_{d \in D} \frac{P(d|G_{k+1})}{P(d|G_k)}$$

$$= \prod_{x\in A,\ y\in B,\ xy \text{ appears in } d\in D} P(x|A)P(y|B)\frac{\sum_{p\in\text{EC-row}} EC(p, d - \text{“xy”})}{EC(\text{“xy”}, d - \text{“xy”})}$$

$$= \prod_{x\in A} P(x|A)^{r_x} \prod_{y\in B} P(y|B)^{c_y} \frac{\prod_{q\in\text{EC-col}} {c'_q}^{c'_q}}{\prod_{\substack{p\in\text{EC-row}\\ q\in\text{EC-col}}} EC(p,q)^{EC(p,q)}} \tag{4}$$

To maximize the likelihood gain, $P(x|A)$ and $P(y|B)$ must take the following form, which can be obtained by applying the Lagrange multiplier method with these two sets of probabilities as the variables.

$$P(x|A) = \frac{r_x}{s} \tag{5}$$

$$P(y|B) = \frac{c_y}{s} \tag{6}$$

where s is the sum of all the cells in BC. This form is also what one would intuitively expect.

Putting it into the likelihood gain formula, we get

$$\max_{P_r} LG(BC) = \prod_{x\in A} \left(\frac{r_x}{s}\right)^{r_x} \prod_{y\in B} \left(\frac{c_y}{s}\right)^{c_y} \frac{\prod_{q\in\text{EC-col}} {c'_q}^{c'_q}}{\prod_{\substack{p\in\text{EC-row}\\ q\in\text{EC-col}}} EC(p,q)^{EC(p,q)}}$$

$$= \frac{\prod_{x\in A} {r_x}^{r_x} \prod_{y\in B} {c_y}^{c_y}}{s^{2s}} \times \frac{\prod_{q\in\text{EC-col}} {c'_q}^{c'_q}}{\prod_{\substack{p\in\text{EC-row}\\ q\in\text{EC-col}}} EC(p,q)^{EC(p,q)}} \tag{7}$$

where P_r represents the set of grammar rule probabilities.

Let a_{xy} be the cell value at row x and column y of the bicluster BC; for the expression-context matrix, let s' be the sum of all values in the matrix, and let r'_p be the sum of row p. Notice that $s = s'$ and $a_{xy} = r'_p$ (where row p of the expression-context matrix represents the symbol pair xy). So we can get

$$\max_{P_r} LG(BC)$$

$$= \frac{\prod_{x\in A} {r_x}^{r_x} \prod_{y\in B} {c_y}^{c_y}}{s^{s}} \times \left(\frac{\prod_{p\in\text{EC-row}} {r'_p}^{r'_p}}{\prod_{\substack{x\in A\\ y\in B}} {a_{xy}}^{a_{xy}}}\right) \times \frac{\prod_{q\in\text{EC-col}} {c'_q}^{c'_q}}{{s'}^{s'} \prod_{\substack{p\in\text{EC-row}\\ q\in\text{EC-col}}} EC(p,q)^{EC(p,q)}}$$

$$= \frac{\prod_{x\in A} {r_x}^{r_x} \prod_{y\in B} {c_y}^{c_y}}{s^{s} \prod_{\substack{x\in A\\ y\in B}} {a_{xy}}^{a_{xy}}} \times \frac{\prod_{p\in\text{EC-row}} {r'_p}^{r'_p} \prod_{q\in\text{EC-col}} {c'_q}^{c'_q}}{{s'}^{s'} \prod_{\substack{p\in\text{EC-row}\\ q\in\text{EC-col}}} EC(p,q)^{EC(p,q)}} \tag{8}$$

It can be seen that the two factors in Eq.8 are of the same form, one for the bicluster and one for the expression-context matrix. Indeed, this form of formula measures the multiplicative coherence of the underlying matrix, in a similar way as in [10]. It reaches the maximum of 1 iff. the underlying matrix has perfect multiplicative coherence (easy to prove by using the Lagrange multiplier method). Therefore we get the conclusion that, by extracting a bicluster, the maximal likelihood gain is the product of the multiplicative coherence of the bicluster and the multiplicative coherence of its expression-context matrix.

Notice that in the above formalization, we don't consider the possibility that some sentences containing xy ($x \in A$, $y \in B$) may have been reduced before we learn this AND-OR group. In that case, when this new AND-OR group is learned, we may get new ways of parsing such sentences, thus increasing their likelihood. So when there's significant ambiguity in the target grammar, the above formulas may not give an accurate estimation of the real likelihood gain.

Now let's turn to the prior, which is solely determined by the grammar size. By extracting a bicluster, a set of new rules are added into the grammar, which has $4+(2+2|A|)+(2+2|B|)$ symbols. On the other hand, each reduction (of xy to N) decreases the grammar size by 4 symbols, and there are $s = \sum_{x \in A, y \in B} a_{xy}$ number of such reductions. So overall,

$$\frac{P(G_{k+1})}{P(G_k)} = \frac{2^{-(DL(G_K)+(4+(2+2|A|)+(2+2|B|))\alpha - 4s\alpha)}}{2^{-DL(G_k)}} = 2^{\alpha(4s-2|A|-2|B|-8)} \quad (9)$$

where α is the number of bits needed to represent a symbol.

Combining Eq.8 and Eq.9, we can get the posterior gain formula when extracting a bicluster (with the optimal grammar rule probabilities assigned), as shown in Eq.2 in the main text. Notice that Eq.5 and 6 still hold for maximizing the posterior gain, because the values of $P(x|A)$ and $P(y|B)$ don't have any effect on the prior gain.

A.2 Attaching the New AND Symbol under Existing OR Symbols

In this section we try to formalize how the attaching step (Section 3.2 in the main text) changes the posterior. Suppose we add a new rule $O \to N$ into the grammar G_k, and do a maximal reduction on the involved sentences, resulting in a new grammar G_{k+1}. Suppose O was learned by extracting the bicluster BC, together with M and P s.t. $M \to OP$ (the following derivation can also be applied to $M \to PO$). So O corresponds to the set of rows of BC and P corresponds to the set of columns. By adding the rule $O \to N$, we expand BC by adding a new row, which records the appearance number of Ny in the corpus for each $y \in P$. Let EC be the expression-context matrix of BC, and EC-row and EC-col be the set of rows and columns of EC. With the new rule $O \to N$, EC is also expanded with a set of new rows for the new expressions containing N, and we use $EC(\text{“Ny”}, q)$ to represent the value at the new row Ny and column q in the expanded expression-context matrix. Because we may change the rule probabilities after adding the new rule, denote the original rule probabilities by $P()$ and the new rule probabilities by $P'()$.

The likelihood is changed in the following way. First, for the sentences involved in the new row for N, each appearance of $Ny\,(y \in P)$ is reduced to M, leading to a likelihood change just as discussed in the previous section. Second, for the sentences involved in BC, since we change the probabilities of rules related to BC, and the reduction of Ny to M results in more equivalent sentences, their likelihood is changed accordingly.

$$\frac{P(D|G_{k+1})}{P(D|G_k)}$$

$$= \prod_{y \in P,\ Ny \text{ appears in } d \in D} \left(P'(N|O)P'(y|P) \frac{\widetilde{col}(d - \text{“Ny”})}{EC(\text{“Ny”}, d - \text{“Ny”})} \right)$$

$$\times \prod_{x \in O,\ y \in P,\ xy \text{ appears in } d \in D} \left(\frac{P'(x|O)P'(y|P)}{P(x|O)P(y|P)} \times \frac{\widetilde{col}(d - \text{“xy”})}{\sum_{p \in \text{EC-row}} \widetilde{EC}(p, d - \text{“xy”})} \right) \tag{10}$$

where $\widetilde{col}()$ is defined as

$$\widetilde{col}(cont) = \sum_{p \in \text{EC-row}} \widetilde{EC}(p, cont) + \sum_{z \in P} EC(\text{“Nz”}, cont) \tag{11}$$

$\widetilde{EC}(p,q)$ represents the value of cell pq in the *derived* expression-context matrix, which is the expected appearance number of the combination of expression p and context q when the current learned grammar G_k is applied to expand the current partially reduced corpus. To construct $\widetilde{EC}$, we have to enumerate all the AND symbols that M may be directly or indirectly reduced to, and traverse their appearances in the partially reduced corpus. Based on the definition of $\widetilde{EC}$, it's obvious that it is perfectly multiplicatively coherent.

Let $\widetilde{EC}'$ be the expanded derived expression-context matrix containing both $\widetilde{EC}$ and the new rows for Ny ($y \in P$). So $\widetilde{col}(q)$ is the sum of column q in $\widetilde{EC}'$. Let EC-row$'$ and EC-col$'$ be the set of rows and columns of $\widetilde{EC}'$. Let EC' be the actual expanded expression-context matrix containing both EC and the new rows. Let $col(q)$ be the sum of column q in EC'.

Let $\widetilde{BC}$ be the derived bicluster that records the expected appearance number of each symbol pair xy ($x \in O$, $y \in P$) when applying the current learned grammar G_k to expand the current partially reduced corpus. So $\widetilde{EC}$ is its expression-context matrix. It can be proved that, when recursive rules are not involved in generating symbol pairs in $\widetilde{BC}$, it has the same row sums, column sums and total sum as BC, but its cell values may be different, which makes it perfectly multiplicatively coherent. When recursive rules are involved, however, $\widetilde{BC}$ and BC might be quite different. Let r_x be the sum of row x and c_y be the sum of column y in $\widetilde{BC}$. Let $\widetilde{BC}'$ be the expanded derived bicluster that contains both $\widetilde{BC}$ and the new row for N. Let r_N be the sum of the new row for N, and a_{Ny} be the cell value at column y in the new row.

Since $P(x|O)P(y|P)\sum_{p \in \text{EC-row}} \widetilde{EC}(p, d - \text{“xy”}) = \widetilde{EC}(\text{“xy”}, d - \text{“xy”})$, we may further reduce Eq.10 as follows.

$$\frac{P(D|G_{k+1})}{P(D|G_k)} = \prod_{x \in O \cup \{N\}} P'(x|O)^{r_x} \prod_{y \in P} P'(y|P)^{c_y + a_{Ny}} \frac{\prod_{q \in \text{EC-col}'} \widetilde{col}(q)^{col(q)}}{\prod_{\substack{p \in \text{EC-row}' \\ q \in \text{EC-col}'}} \widetilde{EC}'(p,q)^{EC'(p,q)}} \tag{12}$$

It can be proved that, if for every AND-OR group involved in calculating $\widetilde{EC}$, the bicluster and its expression-context matrix are both perfectly multiplicatively coherent, and if no recursive rules are involved, then $EC = \widetilde{EC}$. Since we learn new rules only when Eq.2 or Eq.3 is large enough, we expect that the likelihood gain of each step in the algorithm is close to the maximum of 1 and thus the biclusters and their expression-context matrixes are approximately multiplicatively coherent. So we use $\widetilde{EC}$ to approximate EC and use $\widetilde{col}(q)$ to approximate $col(q)$, and therefore according to Eq.4 we get

$$\frac{P(D|G_{k+1})}{P(D|G_k)} \approx LG(\widetilde{BC}') \tag{13}$$

Again it can be shown that for the new set of rule probabilities P_r, Eq.5 and 6 hold when the likelihood gain is maximized. In addition we know both $\widetilde{BC}$ and $\widetilde{EC}$ are perfectly multiplicatively coherent. So we get

$$\begin{aligned}
&\max_{P_r} \frac{P(D|G_{k+1})}{P(D|G_k)} \approx \max_{P_r} LG(\widetilde{BC}') \\
&= \frac{f(r_N) \times \prod_{y\in P} f(c_y + a_{Ny}) \times f(s)^2 \times \prod_{q\in \text{EC-col}} f\left(c'_q + \sum_{y\in P} EC(\text{“Ny”}, q)\right)}{\prod_{y\in P} f(c_y) \times f(s+r_N)^2 \times \prod_{\substack{y\in P \\ q\in \text{EC-col}}} f(EC(\text{“Ny”}, q)) \times \prod_{q\in \text{EC-col}} f(c'_q)}
\end{aligned} \tag{14}$$

where $f(x) = x^x$; s is the total sum of $\widetilde{BC}$; c'_q is the sum of column q of $\widetilde{EC}$.

Notice that there might be a third part in the likelihood change in addition to the two discussed above: after Ny is reduced to M, it may be further reduced, leading to a series of likelihood changes. However, it can be proved that if 1) Eq.14 reaches its maximum of 1, i.e., $\widetilde{BC}'$ and its expression-context matrix are perfectly multiplicatively coherent, and 2) for every AND-OR group involved in calculating $\widetilde{EC}$, the bicluster and its expression-context matrix are both perfectly multiplicatively coherent, then the likelihood gain caused by this part is 1, i.e., the likelihood is not changed. Since we learn new rules only when Eq.2 or Eq.3 is large enough, we expect that both conditions are approximately satisfied and the likelihood change caused by this part is small. Besides, we have to do maximal reduction to calculate the effect of this part, which would be too time-consuming if we do it for every candidate new rule. So we choose to omit this part.

Now let's turn to the prior. There are two changes of the grammar length. First, a new rule $O \to N$ is added into the grammar. Second, we reduce $Ny\,(y \in P)$ to M, so in the grammar, for each appearance of Ny, the two rules that generate N and y are now combined to one that generates M. Therefore the prior gain is

$$\frac{P(G_{k+1})}{P(G_k)} = \frac{2^{-(DL(G_K)+2\alpha-4r_N\alpha)}}{2^{-DL(G_k)}} = 2^{\alpha(4r_N-2)} \tag{15}$$

where r_N is the sum of the new row for N in $\widetilde{BC}'$. Again, here we omit the changes caused by possible further reductions after Ny is reduced to M.

Putting Eq.14 and 15 together, we get the approximate posterior gain when learning a new rule in the attaching step (with the optimal grammar rule probabilities assigned). It's easy to see that the result is equal to the ratio of the maximal posterior gain by extracting $\widetilde{BC}'$ to the maximal posterior gain by extracting $\widetilde{BC}$, as shown in Eq.3 of the main text.

References

1. Adriaans, P., Trautwein, M., Vervoort, M.: Towards high speed grammar induction on large text corpora. In: SOFSEM 2000, LNCS 1963. (2000)
2. van Zaanen, M.: Abl: Alignment-based learning. In: COLING. (2000)
3. Clark, A.: Unsupervised induction of stochastic context-free grammars using distributional clustering. In: Proceedings of CoNLL. (2001)
4. Clark, A.: Learning deterministic context free grammars: The omphalos competition. Machine Learning **66** (2007)
5. Solan, Z., Horn, D., Ruppin, E., Edelman, S.: Unsupervised learning of natural languages. Proc. Natl. Acad. Sci. **102**(33) (August 2005) 11629–11634
6. Chen, S.F.: Bayesian grammar induction for language modeling. In: Proceedings of the 33rd annual meeting on Association for Computational Linguistics. (1995)
7. Kurihara, K., Sato, T.: An application of the variational bayesian approach to probabilistic contextfree grammars. In: IJCNLP-04 Workshop beyond shallow analyses. (2004)
8. Kurihara, K., Sato, T.: Variational bayesian grammar induction for natural language. In: ICGI 2006. Volume 4201 of LNAI. (2006) 84–96
9. Liang, P., Petrov, S., Jordan, M.I., Klein, D.: The infinite pcfg using hierarchical dirichlet processes. In: Proceedings of EMNLP-CoNLL. (2007) 688–697
10. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: A survey. IEEE/ACM Trans. on Comp. Biol. and Bioinformatics **1**(1) (2004) 24–45
11. Stolcke, A.: Boogie. ftp://ftp.icsi.berkeley.edu/pub/ai/stolcke/software/boogie .shar.z (1993)
12. Baker, J.K.: Trainable grammars for speech recognition. In: Speech Communication Papers for the 97th Meeting of the Acoustical Society of America. (1979)
13. Lari, K., Young, S.: The estimation of stochastic context-free grammars using the inside-outside algorithm. Computer Speech and Language **4** (1990) 35–36
14. Klein, D., Manning, C.D.: Corpus-based induction of syntactic structure: Models of dependency and constituency. In: Proceedings of ACL. (2004)
15. Bod, R.: An all-subtrees approach to unsupervised parsing. In: Proceedings of ACL. (2006)