Reproducing Bayesian Model Merging for Stochastic Context Free Grammar Induction

James Karlsson

September 2026

Abstract

Stolcke and Omohundro [8] induce a stochastic context free grammar by starting from a grammar that memorises its training corpus and applying two structural operators, merging and chunking, for as long as a Bayesian posterior over grammars improves. We reimplemented their method in OCaml and report which of their results it reproduces. Our implementation scores a grammar by a description length prior together with an approximate marginal likelihood, which fits the production probabilities with the inside outside algorithm and then applies a Dirichlet correction to the expected production counts, and it also offers a mean field variational bound. Both scores equal the Dirichlet multinomial evidence of the expected counts plus the entropy of the posterior over parses, taken at different parameter settings. They coincide whenever every parse of a string uses the same productions and otherwise differ, by 1.42 nats on a small ambiguous grammar. On the eight example languages of the original paper and three further synthetic ones, every induced grammar generates exactly the target language up to length seven and every recursive target yields a recursive grammar, although the result for the largest target comes from the run committed to the repository because our rerun did not finish. We also measure the absolute frequency effect that the original paper predicts. With counts in the ratio $6:3:1$, the running example $\{ab, aabb, aaabbb\}$ generalises to $a^n b^n$ at 200 samples and stops generalising at 300. For four of the eleven targets the held out sample contains no string missing from the training sample, so on those languages the evidence that the method generalises is the recursive structure of the induced grammar, and the reported held out likelihood cannot show it.

1 Introduction

Learning a probabilistic grammar from strings involves two problems of different kinds. Once the productions are fixed, estimating their probabilities is a continuous problem with a standard solution in the inside outside algorithm of Baker [1], as developed by Lari and Young [6], which is an instance of expectation maximisation [3]. Choosing the productions in the first place is a discrete search over a space of grammars that grows combinatorially with the number of nonterminals and the length of the right hand sides, and likelihood alone cannot guide that search because the most likely grammar just lists the training strings.

Bayesian model merging [8] handles both problems with a single objective. A prior over grammar structures penalises size while the marginal likelihood rewards fit, and the search starts from the most specific grammar, with one production for each distinct training string, and applies local generalising operators for as long as the posterior improves. The original paper recovers the eight test grammars that Cook, Rosenfeld and Aronson [2] used for their hill climbing method, and it also infers the palindromes ww^R over $\{a, b\}$, a language that Cook et al. list as beyond the reach of their algorithm.

We reimplemented the method and ran a series of experiments against it. Section 2 reviews stochastic grammars and the Bayesian formulation, and Section 3 describes the representation, parser, scores, operators and search of our implementation. Section 4 relates the two approximations to the marginal likelihood that the implementation offers, and Section 5 follows the running example of the original paper to measure how the absolute size of the corpus controls generalisation. Section 6 reports results on eleven target languages, and Section 8 sets out the limitations of the implementation and of the evaluation. Every number reported here comes from running the code at commit `ae580bd` of the repository, with one exception that Section 7 explains.

2 Background

2.1 Stochastic context free grammars

A stochastic context free grammar pairs a context free grammar $G = (N, \Sigma, P, S)$ with a probability $\theta_{A \to \gamma}$ for each production, and the probabilities of the productions that share a left hand side A sum to one. A derivation has the product of the probabilities of the productions it uses, and a string has the sum over its derivations. For a string $w = w_1 \cdots w_n$ the inside probability $I(A, i, j)$ is the probability that A derives $w_{i+1} \cdots w_j$, so that $P(w) = I(S, 0, n)$, and the outside probability $O(A, i, j)$ is the probability of deriving $w_1 \cdots w_i \, A \, w_{j+1} \cdots w_n$ from S. The expected number of uses of a production $A \to \gamma$ in the derivations of w is

$$\mathbb{E}[n_{A \to \gamma}] = \frac{1}{P(w)} \sum_{0 \le i < j \le n} O(A, i, j) \, \theta_{A \to \gamma} \, I(\gamma, i, j),$$

where $I(\gamma, i, j)$ is the inside probability of the sequence γ over the span. Expectation maximisation alternates between computing these expected counts over the corpus and setting each probability to its count divided by the total count of its left hand side [1, 6, 3].

2.2 Bayesian model merging

Stolcke and Omohundro [8] separate a grammar M into its structure M_S and its parameters θ_M, and they search for the structure that maximises the posterior

$$P(M_S \mid X) \;\propto\; P(M_S) \int P(\theta_M \mid M_S) \, P(X \mid M_S, \theta_M) \, d\theta_M,$$

where X is the corpus. The structural prior $P(M_S)$ is a description length prior that favours grammars which take fewer bits to write down, and the parameter prior $P(\theta_M \mid M_S)$ is a product of Dirichlet distributions with one factor for each left hand side. The search begins from the grammar with one production for each distinct string in the corpus, which has maximal likelihood and a long description, and it applies two operators to it. Merging replaces two nonterminals by one that has the union of their productions, which generalises the language and usually shortens the description. Chunking replaces a contiguous sequence of symbols by a fresh nonterminal that expands to it, which leaves the language unchanged but can set up a merge that makes the grammar recursive. A chunk on its own usually lowers the posterior, so the original paper searches with a beam instead of merging greedily, which lets a chunk survive long enough for the merge that follows it.

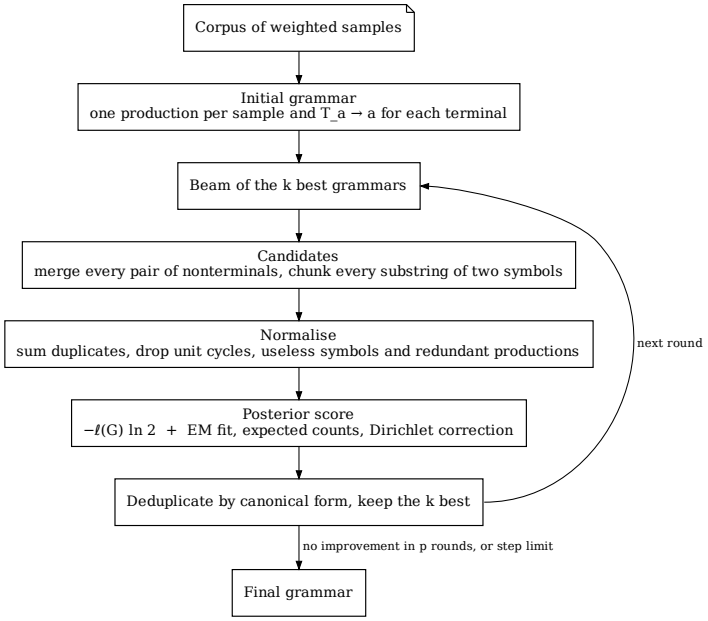

Figure 1: The induction procedure. Each round expands every grammar in the beam by every candidate operator, normalises and scores the results, and keeps the k best distinct grammars, until p rounds pass without improving the best score or the step limit is reached.

3 Implementation

The implementation is an OCaml library with a command line driver. Figure 1 shows how a corpus passes through it, and the subsections below take its stages in the same order.

3.1 Representation and data incorporation

A corpus is a list of distinct token sequences with real valued counts, and a grammar is a start symbol together with a list of productions, each carrying a left hand side, a right hand side of terminal and nonterminal symbols, and a count. Following Section 3.3 of the original paper, the initial grammar introduces a nonterminal T_a with the single production $T_a \to a$ for every terminal a in the corpus, and every distinct training string w with count c_w contributes a production $S \to T_{w_1} \cdots T_{w_n}$ with that count. The structural productions start out containing only nonterminals, which lets merging act on terminal classes in the same way as on any other nonterminal.

3.2 Parsing

The parser computes inside probabilities directly over right hand sides of any length, so grammars never have to be converted to Chomsky normal form. It memoises the inside value of each nonterminal over each span and the inside value of each suffix of each right hand side over each span, and it enumerates the split point between the first symbol of a suffix and the rest. The model has no empty productions, so every symbol covers at least one token, which bounds the enumeration and prevents a nonterminal from recurring on its own span except through unit productions. The inside pass takes time proportional to n^3 multiplied by the total length of the right hand sides.

The outside pass processes spans in decreasing order of length. Within a span it visits nonterminals in a topological order of the unit production graph, so the outside value of a

nonterminal is complete before it propagates through a unit production to its child, and the parser refuses any grammar whose unit production graph contains a cycle. For each occurrence of a nonterminal in a right hand side the implementation sums over the start and end positions of that occurrence within the parent span, which makes the outside pass as written proportional to n^4 multiplied by the total right hand side length. All inside and outside quantities are held as logarithms and combined with a numerically stable log sum. Expectation maximisation runs for at most eight iterations by default and stops early once the corpus log likelihood changes by less than 10^{-7}.

3.3 Structural prior

The description length of a grammar, in bits, is

$$\ell(G) = \log_2 |\mathcal{N}| + \log_2 |\Sigma| + \sum_{A \in \mathcal{N}} \Big[\log_2 |P_A| + \sum_{A \to \gamma \in P_A} \Big(\log_2 |\gamma| + \sum_{x \in \gamma} b(x) \Big) \Big],$$

where P_A is the set of productions of A, $b(x) = \log_2 |\mathcal{N}|$ for a nonterminal and $b(x) = \log_2 |\Sigma|$ for a terminal, and $\log_2 1 = 0$. The original paper only fixes a cost of $\log_2 |\mathcal{N}|$ bits for each nonterminal occurrence, so the remaining terms, which encode the alphabet sizes, the number of productions of each nonterminal and the length of each right hand side, are our own choice. The log prior in nats is $-w\,\ell(G)\ln 2$, where the weight w defaults to one.

3.4 Marginal likelihood

The integral over parameters has a closed form only when each string has a single parse, because the likelihood of an ambiguous string is a sum over its derivations. The default score, which the code calls the approximate evidence, first fits $\hat{\theta}$ by expectation maximisation and computes the expected counts $\hat{c}$ under $\hat{\theta}$, and it then replaces the maximum likelihood value of those counts by their Dirichlet multinomial marginal. For a left hand side A with k productions, expected counts $c_1, \ldots, c_k$ and total c_A, the Dirichlet multinomial log marginal with symmetric concentration α is

$$\log \mathrm{DM}_A(c) = \log \Gamma(k\alpha) - \log \Gamma(c_A + k\alpha) + \sum_{i=1}^{k} \big[\log \Gamma(c_i + \alpha) - \log \Gamma(\alpha) \big],$$

and the approximate evidence is

$$\mathcal{E}_{\mathrm{approx}}(G) = \sum_{w \in X} c_w \log P(w \mid \hat{\theta}) + \sum_{A} \Big[\log \mathrm{DM}_A(\hat{c}) - \sum_{i} \hat{c}_i \log \frac{\hat{c}_i}{\hat{c}_A} \Big].$$

The concentration α defaults to 0.1. A grammar that leaves any training string without a parse scores $-\infty$, which rules out grammars that fail to cover the corpus.

The alternative score is the mean field variational lower bound in which the posterior over parameters is a Dirichlet with pseudocounts $\beta_i = \alpha + \mathbb{E}_q[n_i]$, in the style of Kurihara and Sato [4]. The implementation sets the geometric mean weights $\bar{\theta}_i = \exp\big(\psi(\beta_i) - \psi(B_A)\big)$, where B_A is the sum of the pseudocounts for the left hand side of production i and ψ is the digamma function, runs the inside and outside passes under those weights to obtain expected counts, updates the pseudocounts, and repeats. The bound it reports is

$$\mathcal{E}_{\mathrm{vb}}(G) = \sum_{w \in X} c_w \log Z_w(\bar{\theta}) - \sum_{A} \mathrm{KL}\big(\mathrm{Dir}(\beta_A) \,\big\|\, \mathrm{Dir}(\alpha)\big),$$

where $Z_w(\bar{\theta})$ is the inside value of w under the unnormalised weights. A third mode scores by the maximum likelihood alone and makes no correction for uncertainty in the parameters. The log gamma function uses the Lanczos approximation [5], and the digamma function uses its recurrence together with an asymptotic series.

3.5 Operators and normalisation

The candidate operators on a grammar are a merge of every unordered pair of nonterminals and a chunk of every distinct contiguous pair of symbols that occurs in a right hand side, and an option allows longer chunks. A merge keeps the start symbol if it is one of the pair and otherwise keeps the lexicographically smaller name, so the result does not depend on the order in which the pair was generated. The original paper does not say what happens when a sequence occurs more than once in a right hand side, so a chunk introduces a fresh nonterminal and replaces either every nonoverlapping occurrence of the sequence or only the leftmost one, depending on an option.

Normalisation after each operator combines duplicate productions and sums their counts, removes unit productions that close a cycle, and removes nonterminals that derive no string or cannot be reached from the start symbol, together with their productions. A production is also deleted whenever its left hand side can still derive its right hand side as a sentential form without it. That last check is a chart computation over the sentential form with unit productions closed at each span, and it removes, for example, the production $S \to T_a X T_b$ once $S \to T_a S T_b$ and $X \to T_a T_b$ are both present.

3.6 Search

The default search keeps a beam of width k. Each round applies every candidate operator to every grammar in the beam, normalises and scores each result, discards any result that scores $-\infty$, and keeps the k highest scoring distinct grammars as the next beam. The best grammar so far is replaced whenever the top of the new beam beats it by more than 10^{-9}, and the search stops after a round limit or after p consecutive rounds without such an improvement. Two grammars count as the same when they are equal up to a renaming of nonterminals. The implementation decides this by computing a canonical string for each grammar, which it does by refining a colouring of the nonterminals by the multiset of their production signatures and, where a colour class remains ambiguous, branching on each member of the first such class and keeping the lexicographically smallest encoding. This is the individualisation and refinement scheme used for graph canonisation [7], and its worst case is exponential in the size of the ambiguous classes. The driver also offers a best first search over a bounded frontier, which the experiments below do not use.

4 The two evidence scores

The test suite checks that the two scores of Section 3.4 agree on an unambiguous grammar. The following proposition shows exactly when that agreement holds and why it fails in general.

Proposition 1. *Let $H(r)$ denote the entropy of a distribution r over the parses of the corpus, weighted by the string counts. At a fixed point of expectation maximisation, where $\hat{\theta}_i = \hat{c}_i / \hat{c}_A$,*

$$\mathcal{E}_{\mathrm{approx}}(G) = \sum_A \log \mathrm{DM}_A(\hat{c}) + H(r_{\hat{\theta}}),$$

where $r_{\hat{\theta}}$ is the posterior over parses under $\hat{\theta}$. At a fixed point of the variational iteration,

$$\mathcal{E}_{\mathrm{vb}}(G) = \sum_A \log \mathrm{DM}_A\big(\mathbb{E}_q[n]\big) + H(q),$$

where q is the posterior over parses under the weights $\bar{\theta}$.

Proof. For the first identity, the log likelihood of the corpus equals the expected complete data log likelihood under the parse posterior plus the entropy of that posterior, which gives $\sum_w c_w \log P(w \mid \hat{\theta}) = \sum_i \hat{c}_i \log \hat{\theta}_i + H(r_{\hat{\theta}})$. At the fixed point $\hat{\theta}_i = \hat{c}_i / \hat{c}_A$, so the first term cancels the maximum likelihood term that $\mathcal{E}_{\text{approx}}$ subtracts. For the second identity, $\log Z_w(\bar{\theta})$ decomposes in the same way into $\sum_i \mathbb{E}_q[n_i] \log \bar{\theta}_i + H(q)$. The Kullback Leibler divergence between Dirichlet distributions is $\log B(\alpha) - \log B(\beta) + \sum_i (\beta_i - \alpha)\big(\psi(\beta_i) - \psi(B_A)\big)$, where B is the multivariate beta function, and at the fixed point $\beta_i - \alpha = \mathbb{E}_q[n_i]$ and $\psi(\beta_i) - \psi(B_A) = \log \bar{\theta}_i$, so its last sum equals the first term of $\log Z$. What remains is $\log B(\beta) - \log B(\alpha) + H(q)$, and $\log B(\alpha + c) - \log B(\alpha)$ is $\log \mathrm{DM}(c)$. □

Both scores add an entropy term to a Dirichlet multinomial evidence, and they differ only in the parameters at which the parse posterior and the expected counts are taken. When the corpus is unambiguous under G, both entropies vanish and both sets of expected counts equal the fixed counts of the unique parses, so the two scores are identical. They are also identical whenever all parses of each string use the same multiset of productions, as with $S \to S\,S \mid a$, where the posterior over parses is uniform whatever the parameters. In every other case the scores differ, as Table 1 shows for the grammar $S \to S\,S \mid S\,S\,S \mid a \mid b \mid a\,b$ on the corpus $\{7\colon ab,\ 3\colon aba,\ 2\colon baba,\ 4\colon a,\ 1\colon ababa\}$. The difference there grows with α and changes sign, so the approximate evidence is neither a lower nor an upper bound on the variational bound. All the grammars in the experiments of Section 6 are scored with the approximate evidence.

α	Iterations	$\mathcal{E}_{\text{approx}}$	$\mathcal{E}_{\text{vb}}$	Difference
0.1	8	−50.893	−50.907	+0.014
0.5	30	−47.437	−47.297	−0.141
2.0	30	−47.808	−46.391	−1.417

Table 1: The two scores on an ambiguous grammar, in nats. On the unambiguous running example and on $S \to S\,S \mid a$ the two scores agree to within 10^{-14}.

5 The running example and the effect of corpus size

The running example of the original paper is the corpus $\{ab, aabb, aaabbb\}$. That paper does not reprint its counts, so we use counts of 30, 15 and 5. On the search path shown in Figure 2, the initial grammar G_0 has a posterior of -72.35 nats and a description length of 30.8 bits, and chunking $T_a T_b$ into X lowers the posterior to -74.23, because the grammar gains a production and a nonterminal while still generating the same language. The merge of S with X then raises it to -62.51, and once normalisation has removed the productions that the new recursive production makes redundant, the result is $S \to T_a S T_b \mid T_a T_b$, which generates $a^n b^n$. Figure 2 of the original paper reaches the same grammar in four steps, chunking AB, then chunking AXB, and merging twice. Our redundancy check removes $S \to T_a X T_b$ as soon as the merge makes it derivable, so one chunk and one merge are enough. Expectation maximisation gives the recursive production a probability of $1/3$, the maximum likelihood value for 25 recursive expansions among 75 uses of S in the corpus.

In its discussion of related work, the original paper argues that the absolute number of samples controls how far the method generalises, in addition to their relative frequencies. Multiplying every count by a constant multiplies the likelihood lost by any generalisation by the same constant, while the saving in the prior stays fixed. We tested this by holding the three counts in the ratio $6 : 3 : 1$ and scaling the total, and Table 2 gives the results. The corpus generalises to $a^n b^n$ at every scale up to 200 samples, with a posterior gain that shrinks as

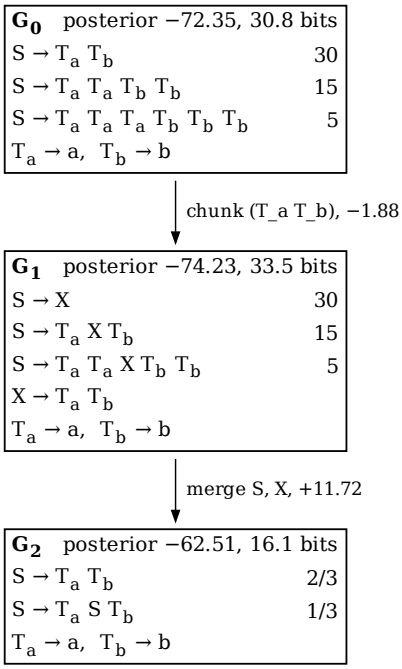

Figure 2: The accepted steps on the running example, with posteriors in nats and description lengths in bits. The counts in G_0 and G_1 are sample counts, and the values in G_2 are the fitted production probabilities. The chunk lowers the posterior by 1.88 nats and the merge raises it by 11.72.

the corpus grows, and from 300 samples upwards the search accepts no step and returns the memorising grammar. For this ratio and the default settings the threshold lies somewhere between 200 and 300 samples.

Counts	Initial posterior	Final posterior	Recursive
6, 3, 1	−34.78	−23.51	yes
15, 7, 2	−47.28	−36.16	yes
30, 15, 5	−72.35	−62.51	yes
60, 30, 10	−117.94	−110.60	yes
120, 60, 20	−208.43	−206.42	yes
180, 90, 30	−298.63	−298.63	no
300, 150, 50	−478.74	−478.74	no
3000, 1500, 500	−4521.79	−4521.79	no

Table 2: Generalisation on the running example as the corpus grows with fixed proportions. The second row rounds the proportions down to integer counts.

The shape of the distribution matters as well as its size. With the near uniform counts 17, 17 and 16, which also total 50, the search accepts no step at all. The recursive grammar gives string lengths a geometric distribution, so its likelihood on the uniform corpus is -68.62 nats against -47.74 on the decaying one, and that loss is larger than what the shorter description saves.

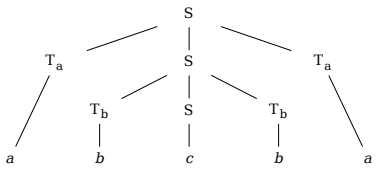

Figure 3: The Viterbi parse of $abcba$ under the grammar induced for the palindromes.

6 Experiments

6.1 Setup

The experiments use eleven target grammars, of which eight are the test grammars in Table 1 of the original paper, taken there from Cook et al. [2], and three are synthetic languages chosen to test nesting and operator precedence. Following that table, the palindrome target is wcw^R with a centre marker, which is easier than the unmarked ww^R. For each target the driver samples 50 training strings and 200 held out strings from the target with every production equally likely, using seeds 1 and 1001, discards strings longer than seven symbols, and combines repeated strings into counts. Induction starts from the initial grammar of the training sample and uses the approximate evidence with $\alpha = 0.1$ and prior weight one, a beam of three, at most 25 rounds and a patience of five rounds, and once the search ends the production probabilities of the final grammar are refitted on the training sample. The original paper trained on the small sets of high probability strings used by Cook et al., with counts scaled to sum to 50, and chose best first search or a beam of width three or four separately for each language. Our setup therefore matches its corpus size without matching its corpora or its search settings.

We compare each induced grammar with its target in three ways. The language comparison enumerates every string of length at most seven that each grammar generates and records whether the induced grammar generates every string of the target, what fraction of its own strings belong to the target, and whether the two sets are equal. The held out log likelihood is the average log probability per token of the held out strings, and it is $-\infty$ if any held out string has no parse. The third check records whether the induced grammar is recursive, which matters because recursion is what allows a grammar to generate strings longer than any it saw in training.

6.2 Results

Table 3 gives the outcome for every target, and every induced grammar parses every held out string, generates exactly the target language up to length seven with a precision of one, and is recursive precisely when its target is. The search settles within one to seven accepted steps, and the posterior improves in every case. The largest improvements come on the arithmetic, basic English and nested parentheses targets, whose training samples contain the most distinct strings and whose initial grammars are consequently the longest.

Several induced grammars are the target grammar up to a renaming of nonterminals. The palindromes give $S \to c \mid T_a S T_a \mid T_b S T_b$, the running language gives $S \to T_a S T_b \mid T_a T_b$, and the nested parentheses give $S \to a \mid T_(\, S \, T_) \mid S \, S$. Figure 3 shows the most probable parse of $abcba$ under the induced palindrome grammar, which has the same nested structure as the target.

The arithmetic target shows that two grammars can generate the same language with different structure. The target separates sums from products with the productions $S \to S + T \mid T$, $T \to T * F \mid F$ and $F \to (S) \mid a \mid b$, whereas the induced grammar is $S \to S \, T_* \, S \mid T_(\, S \, T_) \mid a \mid b$ with

Group	Target	$\lvert\mathcal{N}\rvert$	$\lvert P\rvert$	Initial	Final	Held out	Novel	Steps	Exact	Rec.
paper	parens	3	5	−124.81	−100.91	−0.438	2.5%	2	yes	yes
paper	a2n	2	3	−60.67	−54.43	−0.290	0%	2	yes	yes
paper	abn	3	4	−68.02	−58.54	−0.290	0%	2	yes	yes
paper	anbn	3	4	−76.14	−68.73	−0.336	0%	2	yes	yes
paper	palindrome	3	5	−198.88	−120.80	−0.717	7.5%	1	yes	yes
paper	addition	4	7	−203.57	−134.30	−1.120	20.5%	2	yes	yes
paper	shape	5	7	−251.23	−140.65	−0.646	12.0%	4	yes	yes
paper	basic_english†	11	22	−470.39	−286.80	−0.986	5.0%	7	yes	no
synthetic	nested_parens	3	5	−267.46	−149.87	−0.794	8.5%	1	yes	yes
synthetic	nested_anbn	3	4	−93.67	−68.16	−0.481	0%	1	yes	yes
synthetic	expr	4	8	−424.16	−223.34	−1.291	24.5%	3	yes	yes

Table 3: Results for each target. $\lvert\mathcal{N}\rvert$ and $\lvert P\rvert$ are the sizes of the induced grammar, Initial and Final are posteriors in nats, Held out is the average log probability per token, Novel is the share of held out mass made up of strings absent from the training sample, Steps counts accepted operators, and Exact and Rec. record language equality up to length seven and recursion. The row marked † is the result committed in the repository, which our rerun had not reproduced when this version was built.

$T_* \to * \mid +$. The search has merged the two operators into one class and the three levels into one nonterminal, and the result generates the same strings while being ambiguous and encoding no precedence. With the two languages equal, only the likelihood of the sample can tell the grammars apart, and here the difference in likelihood is too small to outweigh the much shorter description of the collapsed grammar.

The held out figures need the qualification in the Novel column. The languages a^{2n}, $(ab)^n$, $a^n b^n$ and the nested $a^n c\, b^n$ have only three or four strings of length at most seven, and the 50 training samples include all of them, so the 200 held out samples contain no new string and the held out likelihood only measures fit to the training strings. For the same reason, comparing languages up to length seven cannot tell the induced grammar apart from a grammar that memorises the training set on these four targets. What shows that the method generalises on them is that each induced grammar is recursive and so generates strings of every length in the target language. Only the addition and arithmetic targets put a substantial share of held out mass on novel strings, about a fifth and a quarter respectively.

7 Reproducibility

The test suite covers parsing, expected counts, the scores, the operators, normalisation and canonical forms, and it passes in full at commit `ae580bd`. We ran the experiments with the driver's default settings. For the ten targets other than basic English the results match the file `experiments/results.tsv`, committed on 18 September 2026, in every reported column, even though five fixes to parsing, scoring, canonicalisation and evaluation were committed after that file. Each of those ten targets finished within 15 seconds. The basic English target starts from a grammar with 18 nonterminals and 38 productions, and since every round scores every pairwise merge with a full expectation maximisation fit, our rerun was still going after half an hour, so its row in Table 3 is the committed result.

The tool prints production probabilities obtained by normalising the counts carried along the search, and these are different from the fitted parameters that the score uses. On the running example it prints 0.833 for the recursive production, whereas the fitted value given in Section 5 is $1/3$.

8 Limitations

The original paper leaves the description length encoding, the concentration α and the handling of repeated chunk occurrences open, so our results depend on the particular choices made here, and the threshold in Table 2 in particular would move with α and with the prior weight. The default score approximates the marginal likelihood through the expected counts at the maximum likelihood estimate, which Section 4 shows to be exact only for grammars whose parses of each string use the same productions. The variational alternative is a bound, and we have not measured how tight it is.

The evaluation is limited by string length as well, since comparing languages up to length seven checks only a finite part of each language, and for four targets that part is already contained in the training sample. The targets are small, the training samples are drawn from the target grammars themselves with uniform production probabilities, and each experiment uses a single seed. The results therefore show that the method recovers these languages under these conditions and tell us nothing about how it behaves on natural data or on larger grammars. The cost of search grows with the square of the number of nonterminals, because every round scores every pairwise merge with a full expectation maximisation fit, and the canonicalisation that removes duplicates from the beam is exponential in the worst case.

References

[1] J. K. Baker. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1), page S132, 1979. https://doi.org/10.1121/1.2017061

[2] C. M. Cook, A. Rosenfeld and A. R. Aronson. Grammatical inference by hill climbing. *Information Sciences*, 10(1), pages 59 to 80, 1976. https://doi.org/10.1016/0020-0255(76)90061-X

[3] A. P. Dempster, N. M. Laird and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1), pages 1 to 22, 1977. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x

[4] K. Kurihara and T. Sato. Variational Bayesian grammar induction for natural language. In *Grammatical Inference, Algorithms and Applications (ICGI 2006)*, Lecture Notes in Computer Science, pages 84 to 96. Springer, 2006. https://doi.org/10.1007/11872436_8

[5] C. Lanczos. A precision approximation of the gamma function. *Journal of the Society for Industrial and Applied Mathematics, Series B, Numerical Analysis*, 1, pages 86 to 96, 1964. https://doi.org/10.1137/0701008

[6] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4(1), pages 35 to 56, 1990. https://doi.org/10.1016/0885-2308(90)90022-X

[7] B. D. McKay and A. Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60, pages 94 to 112, 2014. https://doi.org/10.1016/j.jsc.2013.09.003

[8] A. Stolcke and S. Omohundro. Inducing probabilistic grammars by Bayesian model merging. In *Grammatical Inference and Applications (ICGI 1994)*, Lecture Notes in Computer Science, pages 106 to 118. Springer, 1994. https://doi.org/10.1007/3-540-58473-0_141, also https://arxiv.org/abs/cmp-lg/9409010