Likelihoods for Minimalist Grammars

Zachary Feldcamp

August 4, 2026

1. The general problem of grammar inference

- In the P&P approach, assumed here, the mind infers a distribution over parameter values from the primary linguistic data (PLD), understood as a set of linguistic observations. The empirical end result is a distribution that has a sharp peak over a group of similar grammars.
- The intended use of "parameter" here is intended to be expansive, characterizing whatever makes grammars of particular languages vary. Many of the parameters specify the individual lexical items.
- *Assumption*: each observation is generated by the same latent grammar (parameter settings).[1]
- Suppose that there are m parameters $\theta_1, \ldots, \theta_m$ and n observations $x_1, \ldots, x_n$. Then the goal is to learn the function $P_{\theta_1,\ldots,\theta_m|x_1,\ldots,x_n}(\cdot|\cdot)$, assigning a *posterior* probability to each set of parameter values conditioned on some particular observations.
- Bayes rule expresses the posterior probability in terms of the *likelihood*, the *prior*, and the *evidence*.
- For a particular choice of parameter values and observations, the posterior probability is given by (1).

(1)
$$P(\theta_1, \ldots, \theta_m | x_1, \ldots, x_n) = \frac{P(x_1, \ldots, x_n | \theta_1, \ldots, \theta_m) P(\theta_1, \ldots, \theta_m)}{P(x_1, \ldots, x_n)}$$

  - The first factor of the numerator on the right-hand side is the *likelihood* of the observations under the given parameterization, measuring goodness-of-fit to the data. In other words, what is the chance that n randomly generated derivations from a grammar with parameter values $\theta_1, \ldots, \theta_m$ will yield observations $x_1, \ldots, x_n$?
  - The second factor in the numerator is the *prior* probability of the particular parameter values $\theta_1, \ldots, \theta_m$. In other words, how much does UG favor such parameter values before having been exposed to the PLD? There may be various kinds of "simplicity" biases, for instance, as is widely assumed.
  - The denominator is the *evidence* term, or the marginal probability of the data, given by $P(x_1, \ldots, x_n) = \sum_{\theta'_1,\ldots,\theta'_m} P(x_1, \ldots, x_n | \theta'_1, \ldots, \theta'_m) P(\theta'_1, \ldots, \theta'_m)$.

- The denominator in (1) can be thought of as a constant, because it does not vary by the choice of parameter values. The equation in (1) can therefore be simply written as the proportionality in (2).

(2)
$$P(\theta_1, \ldots, \theta_m | x_1, \ldots, x_n) \propto P(x_1, \ldots, x_n | \theta_1, \ldots, \theta_m) P(\theta_1, \ldots, \theta_m)$$

[1] In reality, the problem is more complex. The child encounters observations generated by different grammars and must infer a *mixture model* that includes multiple grammars and tells the child the probability that a particular observation was generated by a particular problem. We start instead with the simpler problem of inferring a single grammar.

- Computing the likelihood of the observations comes down to computing the likelihood of the underlying structures that generated the observations:
  - For each observation x_i, compute the set of possible parses or derivations t that yield x_i; compute the probability of each such derivation under a given set of parameter settings:

(3)

$$P(x_1,\ldots,x_n|\theta_1,\ldots,\theta_m) = \prod_{i=1}^{n} \sum_{d:y(d)=x_i} P(d|\theta_1,\ldots,\theta_m)$$

- At the heart of grammar inference, then, is the computation of derivational likelihood $P(t|\theta_1,\ldots,\theta_m)$.
- The form that derivational likelihood takes depends on the nature of the grammar.
  - In the case of Probabilistic Context-Free Grammars (PCFGs), suppose that each θ_r specifies the probability of choosing production rule r and $f_r(t)$ is the number of times that rule r is used in derivation d. We take the product of the probability of each rule used in the derivation:

$$P(d|\theta_1,\ldots,\theta_m) = \prod_r \theta_r^{f_r(t)} \qquad \text{(Johnson et al., 2007)}$$

  - A similar formulation has been produced for Multiple Context-Free Grammars (Hunter & Dyer, 2013), which are weakly equivalent to Minimalist Grammars (Harkema, 2001; Michaelis, 2001).
- For certain generative processes, it is only practical to compute the likelihood for a given hypothesis up to a constant factor. Such unnormalized likelihoods do not permit direct model comparison but may be used in Bayesian inference via Markov Chain Monte Carlo (MCMC) ... more on this next time.
- Let us consider different approaches towards defining likelihoods for MG derivations.

2. Unnormalized likelihoods for Minimalist Grammars

- We would like to define likelihoods for MG derivations, which are trees formed by the closure of the lexicon under Merge and Move.
- Viewed differently, we would like to define a prior over MG derivations.
- A Minimalist Grammar (MG) G has the form $G = \langle \Sigma, \text{Syn}, \text{Lex}, F \rangle$ (Stabler & Keenan, 2003).
  - Σ is an alphabet over which phonemic (or graphemic) representations are formed. This is not really important to us, as we are not (presently) considering a generative model of phonological words.
  - The syntactic features Syn are made up of disjoint sets of Sel(ectors) and Lic(ensors).
    - Sel is a set of "selector" features (e.g., $[= f]$) or categorial (selectee) features (e.g., $[f]$) related to (external) merge.
    - Lic is a set of "licensing" features related to movement, made up of probe features (e.g., $[+f]$) and goal features (e.g., $[-f]$).
  - The Lex(icon) is a subset of $\Sigma^* \times F^*$.
  - F consists of the structure building functions MERGE and MOVE.
  - The language produced by a MG is closure under the two operations Merge and Move.
- We would like to define a probability distribution over the trees produced by a given MG. We consider two approaches:

– *Bottom-up*. A MG tree is formed by a stochastic process of (i) randomly selecting a term (either a lexical item or a tree) and emitting it, or (ii) randomly selecting two terms and combining them via Merge. Move applies deterministically in the right structural configuration.
– *Top-down*. A MG tree is formed by a stochastic process of starting from a given start symbol (e.g., CP), randomly selecting a lexical head (e.g., C), and recursively generating a compatible sister to the head.

2.1. A prior over bottom-up MG derivations

- Define $P : G \to \mathbb{R}$ as in (4). The function P is well-defined because every syntactic object is either a lexical item (A), a tree formed by Merge (B) of two sisters with matching selector ($= \beta$) and category (β) features, or (C) a tree formed by Move applied to a tree containing both a licensor ($+\beta$) and a licensee ($-\beta$).

(4) (A) $P(l) = \frac{1}{|Lex|}$
(B) $P(\text{MERGE}([\alpha := \beta, \ldots], [\alpha' : \beta, \ldots])) = \frac{1}{\lambda} P([\alpha := \beta, \ldots]) \cdot P([\alpha' : \beta, \ldots])$
(C) $P(\text{MOVE}([\alpha : +\beta, \ldots, -\beta, \ldots])) = P([\alpha : +\beta, \ldots, -\beta, \ldots])$

  – Part (A) of the definition encodes the hypothesis that the probability of observing a particular lexical item should be inversely proportional to the number of potential alternative lexical items (cf. the "size principle" for scoring hypotheses Tenenbaum, 1999).[2]
  – Parts (B) and (C) recursively define the probability of a tree in terms of its immediate subtrees. Part (B) imposes a penalty of $\frac{1}{\lambda}$ for adding additional complexity to a tree via Merge. This is a critical component in being able to make P into a probability distribution. No penalty is imposed on Move because it may only apply finitely many times (as long as there are matching licensor and licensee features).

- To make P into a probability distribution, it suffices to choose λ such that, for some positive real number Z, we have $\sum_{t \in G} P(t) \leq Z$. Then the choice of $\frac{1}{Z}$ as a normalizing constant will give us a probability mass function $f(t) : T \to [0, 1]$.
- Let us consider how to do this.
- For a given MG grammar G and a derivation tree t, let:
  – L : maximum number of licensee features for lexical item.[3]
  – $\ell = \ell(t)$: number of lexical items/leaves in t.
  – $b = b(t)$: number of binary nodes in t (i.e., number of applications of Merge).
  – $u = u(t)$: number of unary nodes in t (i.e., number of applications of Move).
  – $N = N(t)$: number of total nodes in the tree (i.e., $N = \ell + b + u$)
- Also define:
  – T_n : number of labeled MG derivation trees with n lexical items, where we don't care about the identity of the particular lexical items.
  – S_n : number of unlabeled MG tree "shapes" with n lexical items, where we don't care about the identity of the particular lexical items.

[2] In reality, lexical items appear to follow a Zipf distribution. I choose not to encode this into the model because we are interested in comparing different lexicons, not in modeling lexical frequencies *per se*.

[3] In practice, syntactic analyses rarely require L to be larger than 2 or 3.

- For a tree t with n lexical items, we have

  (5) $P(t) = \frac{1}{|Lex|^n \lambda^{n-1}}$,

  because there must have been $n-1$ Merge operations, and Move operations are not penalized.[4]

- Then we have

  (6)

$$\begin{aligned}
\sum_{t \in G} P(t) &= \sum_{n=1}^{\infty} \sum_{t:\ell(t)=n} P(t) \\
&= \sum_{n=1}^{\infty} \sum_{t:\ell(t)=n} \frac{1}{|Lex|^n} \frac{1}{\lambda^{n-1}} \\
&= \sum_{n=1}^{\infty} |\{t : \ell(t) = n\}| \cdot \frac{1}{|Lex|^n} \frac{1}{\lambda^{n-1}} \\
&\leq \sum_{n=1}^{\infty} (T_n \cdot |Lex|^n) \cdot \frac{1}{|Lex|^n} \frac{1}{\lambda^{n-1}} \\
&= \sum_{n}^{\infty} \frac{T_n}{\lambda^{n-1}} \\
&= \lambda \sum_{n}^{\infty} \frac{T_n}{\lambda^n}.
\end{aligned}$$

- To make this sum converge, we can find constants $B, C > 0$ such that $T_n \leq CB^n$, so that

  (7)

$$\begin{aligned}
\sum_{t \in G} P(t) &\leq \lambda \sum_{n=1}^{\infty} C \frac{B^n}{\lambda^n} \\
&= \lambda C \sum_{n=1}^{\infty} \left(\frac{B}{\lambda}\right)^n.
\end{aligned}$$

- Then the sum converges whenever $\lambda > B$ (say, at $\lambda = B+1$). Let us find such a B next.[5]
- We can bound the number of nodes N of a MG tree in terms of the number of leaves n. If a tree has b binary-branching nodes and u unary branching nodes, then we have:

[4] The claim here is that a tree with n leaves (i.e., lexical items) must have involved $n-1$ applications of Merge. This can be proven by induction:

- If there is only one leaf, then there cannot have been any applications of Merge.
- Now suppose that the proposition is true for all trees with n leaves, and consider a tree t with $n+1$ leaves.
- Suppose, moreover, that the root of t is binary-branching (if it is unary-branching, it was not formed by an application of Merge, so we can assume that t is the t top-most node of t that is binary-branching).
- If t is formed by Merge of a leaf with a non-leaf of n leaves, then it took $(n-1)+1 = n$ Merge operations in total.
- Otherwise, it is formed by Merge of two trees, one of m leaves, and the other of $n+1-m$ leaves.
- Then t must have been formed by $(m-1)+((n+1-m)-1)+1 = n$ applications of Merge.
- Then the proposition is true in general.

[5] Presumably, a tighter bound on T_n can be found using methods from analytic combinatorics (e.g., Flajolet and Sedgewick, 2009). Here we are merely interested in finding *some* bound, not necessarily a tight one.

(8)

$$\begin{aligned} N &= b + u + n \\ &= (n-1) + u + n \\ &\leq (n-1) + Ln + n \\ &\leq 2n + Ln \\ &= (L+2)n \\ &=: Kn, \end{aligned}$$

because there are $n-1$ applications of Merge, and at most Ln applications of Move.

- In general, MG trees involve both binary- and unary-branching nodes, formed by Merge and Move respectively. The number of unary-binary trees (Motzkin trees) of n nodes is given by the nth Motzkin number $M_n \leq 3^N$ (Donaghey & Shapiro, 1977; Flajolet & Sedgewick, 2009).
- It follows that every tree "shape" counted in S_n is a Motzkin tree of some size $N \leq Kn$. Then we can bound S_n as follows:

(9)

$$\begin{aligned} S_n &\leq \sum_{N=1}^{Kn} M_N \\ &\leq \sum_{N=1}^{Kn} 3^N \\ &= \frac{3^{Kn+1}-1}{2} \text{(using formula for geometric series)} \\ &\leq \frac{3^{Kn+1}}{2} \\ &= \frac{3}{2} 3^{Kn} \\ &= C(3^K)^n \end{aligned}$$

- Let us now work on bounding T_n.
- We can bound the number of internal-node labels L_{int} by $L_{\text{int}} \leq |\Phi|^{f_{\max}}$, where Φ is the set of features and $f_{\max}$ is the maximum number of features per syntactic head.
- Then we have

(10)

$$\begin{aligned} T_n &\leq S_n \cdot {L_{\text{int}}}^{Kn} \\ &\leq C(3^K)^n \cdot {L_{\text{int}}}^{Kn} \\ &= C\left(3^K {L_{\text{int}}}^K\right)^n. \end{aligned}$$

- So letting $B = (3L_{\text{int}})^K$, we have $T_n \leq CB^n$ for all $n \geq 1$.
- Then letting $Z = \lambda C \sum_{n=1}^{\infty} \left(\frac{B}{\lambda}\right)^n$, we have the following probability mass function:

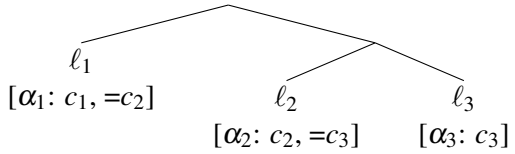
(11) $f(t) = \frac{1}{Z} P(t)$.

- Thus, it is possible in the bottom-up view of structure building to define a probability distribution over MG trees, from which unnormalized likelihoods may be computed for derivations (and for strings, in conjunction with a parser).

Desirable properties of the bottom-up prior

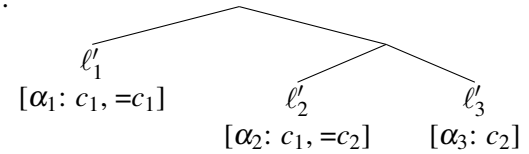
- The bottom-up prior just defined has some desirable properties.
- First, normalization ensures that if a grammar can generate only one derivation, then that derivation has probability 1.
- Conversely, if a grammar can generate many different derivations, then any particular derivation has a relatively small probability.
- Moreover, all else being equal, if Grammar A can generate fewer derivations than Grammar B, then Grammar A is more likely to have generated any particular observation than Grammar B.
- To illustrate, suppose that we observe the string $\alpha_1\alpha_2\alpha_3$ and we are interested in comparing two compatible grammars each with three lexical items.
- In Grammar A, the three lexical items are illustrated in (12), which is also the only tree that can be formed. Even though $P(t) < 1$, we have $f(t) = 1$, because, here, $P(t) = \sum_t P(t)$.

(12) t:



- In contrast, lexical items ℓ'_1 and ℓ'_2 of Grammar B have the same features and are thus interchangeable. Thus t'_1 in (13) exists alongside t'_2 formed by swapping ℓ_1 and ℓ_2. Then we have $f(t'_1) = \frac{P(t'_1)}{P(t'_1)+P(t'_2)} = \frac{P(t'_1)}{2P(t'_1)} = \frac{1}{2}$.

(13) t'_1:



- The likelihood ratio test may then be used to compare between the two grammars: $\frac{f(t)}{f(t'_1)} = 2$. In other words, Grammar A is twice as likely to have generated the string $\alpha_1\alpha_2\alpha_3$ as Grammar B.
- Suppose that two alternative grammars may both produce an infinite number of derivations. One grammar may still be more likely than another by virtue of being capable of producing fewer derivations of a given size bounded by the observed data.

Undesirable properties of the bottom-up prior

- One potentially undesirable property of the bottom-up structural prior is that it corresponds to a generative process that may "crash," or fail to produce a well-formed object. But this is simply a property of the system of Chomsky (1995).
- One practical drawback of the bottom-up approach is that it is generally intractable to compute $Z(G)$, so that, whereas the unnormalized likelihoods $P(t)$ are easy to compute, the true, normalized likelihoods $f(t)$ are not.
- Is $P(t)$ sufficient to do MCMC?

References

Chomsky, N. (1995). *The Minimalist Program*. The MIT Press.

Donaghey, R., & Shapiro, L. W. (1977). Motzkin Numbers. *Journal of Combinatorial Theory*, *23*, 291–301.

Flajolet, P., & Sedgewick, R. (2009). *Analytic Combinatorics*. Cambridge University Press.

Harkema, H. (2001). A Characterization of Minimalist Languages. In *Lacl 2001, lnai 2099* (pp. 193–211).

Hunter, T., & Dyer, C. (2013). Distributions on Minimalist Grammar Derivations. *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, 1–11.

Johnson, M., Griffiths, T. L., & Goldwater, S. (2007). Bayesian Inference for PCFGs via Markov chain Monte Carlo. *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*.

Michaelis, J. (2001). Derivational Minimalism is Mildly Context–Sensitive. *Lecture Notes in Artificial Intelligence*, *2014*, 179–198.

Stabler, E., & Keenan, E. L. (2003). Structural similarity within and among languages. *Theoretical Computer Science*, *293*, 345–363.

Tenenbaum, J. B. (1999). Rules and Similarity in Concept Learning. *Advances in Neural Information Processing Systems 12*, 59–65.