Language Acquisition as Statistical Inference

Mark Johnson
Macquarie University

Preprint version of June, 2013

1 Introduction

Probabilistic grammars define a set of well-formed or grammatical linguistic structures, just as all grammars do. But in addition probabilistic grammars also define probability distributions over these structures, which a statistical inference procedure can exploit. This paper describes two simple statistical inference procedures for probabilistic grammars and discusses some of the challenges involved in generalising them to more realistic settings.

The first three sections of this paper review important ideas from statistical estimation, namely the Maximum Likelihood Principle and Bayesian estimation, and show how they can be applied to learn the parameters of a Probabilistic Context-Free Grammar from surface strings alone in an idealized "toy" scenario. We show that they succeed in learning both the phrase structure and lexical entries without explicit negative evidence in a situation that seems highly challenging for any "staged" learner, i.e., a learner that learns either lexical entries or phrase structure rules first. Then we discuss attempts to generalize these kinds of methods to more realistic scenarios, and point out some of the challenges that are involved in doing this.

One of the main themes of this paper is how little we know about the the linguistic implications of the various approximations made by current statistical inference algorithms. In part because of this, much of the work in this field takes an experimental, "try it and see" approach. The presentation in this paper is deliberately informal: for a more mathematical presentation of much of this material see e.g., Geman and Johnson (2004). The software used to produce the results below is available from http://web.science.mq.edu.au/~mjohnson.

2 Probabilistic Context-Free Grammars

Context-Free Grammars (CFGs) are extremely simple models of constituent structure. It's well-known that simple CFGs are poor descriptions of syntactic structure (Chomsky, 1957), but their simplicity makes them useful for discussing basic ideas of probabilistic models and statistical inference. As explained below, many of these ideas generalize to the more general class of Maxium Entropy or log-linear models, which can be used to define probability distributions over virtually any kind of syntactic structure (Abney, 1997).

Informally, a *Context-Free Grammar* (CFG) is specified in terms of a set of *productions* or *rules* of the form $A \rightarrow B_1 \ldots B_n$. If the grammar contains a rule $A \rightarrow B_1 \ldots B_n$ then a node labeled A is permitted to immediately dominate a sequence of nodes labeled $B_1, \ldots, B_n$ in phrase-structure trees generated by that grammar. CFGs are "context-free" in the sense that the choice of rule is only constrained by the category A, which means that CFGs can only capture longer-range

θ	Rule	θ	Rule
0.8	S $\rightarrow$ NP VP	0.2	S $\rightarrow$ VP
0.1	VP $\rightarrow$ NP V	0.9	VP $\rightarrow$ V NP
0.5	NP $\rightarrow$ *cats*	0.5	NP $\rightarrow$ *dogs*
0.4	VP $\rightarrow$ *chase*	0.6	VP $\rightarrow$ *race*

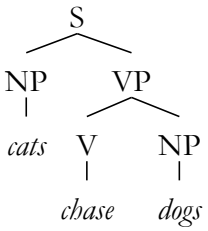

Figure 1: An example of a PCFG and a tree that it generates. This PCFG generates the tree with probability $0.8 \times 0.5 \times 0.9 \times 0.4 \times 0.5 = 0.072$.

dependencies if they decompose into a sequence of local dependencies that are encoded in the nonterminal categories somehow. Gazdar et al. (1985) showed how to do this for a range of English constructions, but CFGs are not capable of describing all natural languages (Shieber, 1985).

A *probabilistic context-free grammar* (PCFG) consists of a CFG together with a *rule probability* $\theta_{A \rightarrow \beta}$ for each rule $A \rightarrow \beta$ in the grammar. These rule probabilities must satisfy the standard constraints of any probability distribution: (i) they must be non-negative, and (ii) the probabilities of all rules expanding each nonterminal must sum to 1.

Figure 1 presents a simple PCFG and a tree that it generates. Linguistically speaking, these PCFG rules encode both the allowable phrase structure configurations (e.g., head-complement orderings) and the syntactic categories of the lexical items.

A PCFG defines a probability distribution over the trees that the CFG generates: the probability of a tree is the product of the probabilities of the rules used to generate it. PCFGs strengthen the "context-freeness" assumption of CFGs: in a PCFG the probability of choosing a rule to expand a node only depends on the category labelling that node, and is independent of the expansions of the other nodes in the tree. Thus just as in CFGs, non-local statistical dependencies can only be captured in a PCFG if they are broken down into a sequence of local dependencies encoded in the nonterminal labels.

Because a PCFG assigns probabilities to trees, it also indirectly assigns probabilities to strings of terminal symbols: the probability of a string of terminals is the sum of the probabilities of all trees that have that string as their yield.

The probabilities $\theta_{A \rightarrow \beta}$ are *parameters* of the PCFG, and they determine the probabilities of the trees that the grammar generates. The fact that $\theta_{\text{VP} \rightarrow \text{V NP}}$ is larger than $\theta_{\text{VP} \rightarrow \text{NP V}}$ means that trees in which a V precedes an NP in a VP will be more probable than trees in which an NP precedes a V in a VP (all else being the same).

PCFG rule probabilities differ from the parameters of the Principles and Parameters approach (Chomsky, 1986) in that rule probabilities are continuously variable (i.e., they range from zero to one) rather than discrete. It's reasonable to be skeptical whether rule probabilities play a causal role in human language: while we can estimate the probability that *dogs* is a noun in a particular corpus, it's not clear it is meaningful to speak of the probability that *dogs* is a noun in English, independent of context, genre, register, etc. Of course it's possible that humans use a statistical learning procedure to identify the rules and lexical items in their language, and simply ignore the associated probabilities once learning is complete (e.g., all rules with an estimated probability less than, say, 10^{-10} might be assumed not to be in the language).

Another problem with PCFGs is that it's not linguistically realistic to associate parameters with specific phrase-structure rules. For example, a head-direction parameter, which specifies whether heads are phrase-initial or phrase-final, would seem to be linguistically more reasonable rather than a separate parameter associated with each phrase-structure rule (although Fodor and Crain (1987)

argue that parameters should be associated with finer-grained details of grammar). In fact, it's relatively easy to modify PCFGs so they associate parameters with sets of rules rather than individual rules (Headden III et al., 2009), so we could easily define a single head-direction parameter that encompasses several phrase-structure rules. However, for simplicity we'll continue to associate parameters with individual phrase-structure rules here.

A very important advantage of continuously-variable parameters is that they permit at least part of language acquisition to be recast as a continuous optimization problem. As Nocedal and Wright (2006) point out, continuous optimization problems are typically easier to solve than discrete optimization problems because continuity provides information about the function's behavior in a neighborhood, and the absence of this information in discrete optimization can be viewed as an extreme case of non-convexity. Associating probabilities with rules enables us to apply statistical procedures for estimating or learning these rule probabilities from data (e.g., a corpus of phrase structure trees or strings) that take advantage of continuity.

3 The Maximum Likelihood Principle

It's natural to ask whether it's possible to estimate the values of the rule probabilities from data in some way. That is, we imagine that we are given a CFG and some data, and our task is to estimate the probabilities associated with the rules in that grammar. Note that this parameter estimation problem can be viewed as a simplified language acquisition problem. For example, in a strongly head-initial language we'd expect estimates of $\theta_{\text{VP}\rightarrow\text{NP V}}$ to be close to zero, and in a strongly head-final language we'd expect estimates of $\theta_{\text{VP}\rightarrow\text{V NP}}$ to be close to zero. Thus learning the rule probabilities of a general PCFG can effectively involve learning important typological properties of a language.

Perhaps unsurprisingly, the complexity of estimating PCFG rule probabilities depends on the kind of data available to the learner. In the simplest version of the problem the learner is provided with a sequence of labeled phrase-structure trees generated according to the unknown rule probabilities, and has to estimate the rule probabilities from this sample of trees. This is called *supervised learning* from *visible data* because the full derivational structure of the sentences is provided (i.e., is visible) to the learner, and is relatively trivial for PCFGs. For example, given phrase-structure trees it is simple to identify the head-complement ordering; simply "read off" the ordering from the trees.

The problem becomes more interesting, and much harder, when the learner is only provided with the yield (i.e., the words labeling the terminal nodes) of the trees, rather than the trees themselves. This is called *unsupervised learning* from *hidden data* because the derivational structure is not provided to (i.e., is hidden from) the learner.[1] The hidden data learning problem for PCFGs is arguably a cognitively more realistic problem, and hence more interesting.

There are many statistical methods one can use to learn PCFG rule probabilities. Here we focus on a very general and powerful method based on the *Principle of Maximum Likelihood* (ML). The ML principle says that to learn or *estimate* the values for parameters Θ from data, one should select values for Θ that make the data as likely as possible. This principle is reasonably intuitive: given a class of models that could possibly explain some data, the Maximum Likelihood Principle instructs us to select the model that makes the data as likely as possible. A learner which follows the ML principle is called a *Maximum Likelihood Estimator* (MLE).

[1] There are intermediate cases — known as *semi-supervised learning* from *partially hidden data* — where the learner is provided with some information about the derivational structure, but not enough to completely determine the phrase-structure tree. For example, Pereira and Schabes (1992) study the interesting situation where the learner observes some but not all of the phrase boundaries and show that the techniques described below generalize to this case.

The ML principle is applicable to a very wide range of estimation problems (i.e., it is not language-specific), and it has a number of attractive theoretical properties. Important among these, it's possible to prove that in a wide range of cases ML estimates are *consistent*, i.e., given data generated by a PCFG, a ML estimator converges on the distribution over trees generated by that PCFG. ML estimates are also *asymptotically optimal* in the sense that there is no other estimator that converges faster for all possible PCFGs.

Perhaps as importantly, ML estimators behave reasonably when their input data does not in fact come from the class of models that the parameters can express, e.g., when the learner is exposed to a language that no parameter setting generates. In this case it's obviously not possible for a parameter-setting learner to learn the target language, but an ML estimator will converge on a model in its model class which generates a language that is as close as possible to the input data (in the sense of Kullback-Leibler divergence, which measures the "distance" between probability distributions). Thus an ML learner using a model class that is much simpler than the one that generated its input data might none the less succeed in learning a useful approximation to the true model. For example, a ML PCFG estimator might be able to learn something useful about the phrase structure of a language even if that language is not generated by a PCFG.

Thus a ML estimator can be viewed as an "ideal observer" kind of learner, in that it most effectively exploits the information available to it. If a ML estimator fails to learn in some situation, then we can be fairly confident that either the model class (e.g., the CFG) being used does not capture the relevant generalizations or the input does not contain sufficient information.

Now PCFGs and the ML principle are what Marr (1982) called "computational level" models; they specify what should be computed, but they do not specify how that computation should be performed. Of course algorithms are important: if human learners actually follow the ML principle they will need an algorithm to estimate the rule probabilities, and we theorists need algorithms for computing ML estimates to find out just what rule probabilities the ML principle predicts.

It turns out that computing the ML estimate for PCFG rule probabilities is trivial in the visible data case (i.e., when the input data consists of phrase-structure trees): it only requires the ability to count and divide, and this can be done incrementally (i.e., as each each tree is observed).

ML estimation is computationally more challenging in the hidden data case, and the algorithms required are much more involved. Unfortunately there are unlikely to be any efficient exact algorithms for ML estimation of PCFGs from strings alone (Cohen and Smith, 2012), so we use methods that approximate the ML estimate. However we don't know how accurate these approximations are, and more importantly, whether these approximations influence the kinds of linguistic generalisations that the estimators can learn.

Because ML estimation is a kind of optimization, standard numerical optimization algorithms can be used here. Alternatively, *Expectation Maximization* (EM) is a general framework for generalizing visible-data learners to hidden-data problems. EM algorithms function by using a current estimate of the parameters to estimate the hidden structures associated with the input data — in this case, this means parsing the input strings using the current estimate of the PCFG rule probabilities — and then inferring the unknown parameters from the estimates of the hidden structures. Computationally the most onerous aspect of the EM algorithm for PCFGs is parsing the input data (updating the PCFG rule probabilities is usually trivial in comparison), suggesting the attractive possibility that at least some aspects of language acquisition may be an almost cost-free by-product of parsing. That is, the child's efforts at language comprehension may supply the information they need for language acquisition.

The rest of this section contains an example of the EM algorithm for computing rule probabilities in PCFGs. EM algorithms are typically batch algorithms that make multiple passes over the input data in order to extract all the information it contains, but recently incremental versions of

S → NP VP	NP → Det N	NP → N Det
VP → V NP	VP → NP V	VP → V NP NP
VP → NP NP V		
Det → *the*	N → *the*	V → *the*
Det → *a*	N → *a*	V → *a*
Det → *cat*	N → *cat*	V → *cat*
Det → *dog*	N → *bone*	V → *bone*
Det → *bites*	N → *bites*	V → *bites*
Det → *gives*	N → *gives*	V → *gives*
the dog bites a cat	*the cat bites a dog*	*a cat gives the dog a bone*
the dog gives a cat the bone	*a dog bites a bone*	

Figure 2: The PCFG rules and input strings used with the Expectation-Maximization and Variational Bayes algorithms, as described in the text. Notice that the grammar is written so that heads can both precede or follow their complements, and the rules introducing terminals (i.e., the lexicon) are completely ambiguous in terms of word to category association. Learning the rule probabilities identifies a subset of these rules that suffice to generate these input strings.

the EM algorithm have been developed (Neal and Hinton, 2001). We use the iterative batch version of the EM algorithm described in Lari and Young (1990) here because it is simple and has few adjustable parameters. On-line learning algorithms typically have additional adjustable parameters (such as learning rates) that affect their estimates in ways that are difficult to understand theoretically. While it's unlikely that human learners use a batch algorithm such as EM,[2] if we're interested in studying ML estimation of PCFGs (e.g., to determine which aspects of language can be learned using ML) what matters are the models which maximize the likelihood, and not the algorithms used to find them.[3]

Figure 2 shows the rules of the PCFG and the input strings used to produce the results below. This grammar massively over-generates: it contains rules for both head-initial and head-final phrase structures, and rules that associate every word with every syntactic category, i.e., all possible lexical entries for the words in this corpus. Linguistically, learning this PCFG's rule probabilities corresponds to learning the head-complement ordering in the language, and to learning the syntactic categories of each lexical item. After learning, the probabilities of the rules corresponding to the unused head-complement ordering and the unused lexical items will be very close to zero, so this approach to learning can be viewed as starting with a wide range of possible structures and lexical entries (specified by Universal Grammar) and "unlearning" the ones that are not used in the language.

Now learning the PCFG rule probabilities would be easy if the learner were given visible input data, i.e., the true phrase-structure trees for sentences in the input data. Moreover, if the learner were told the lexicon (i.e., which syntactic categories are associated with each word) learning the PCFG rules that encode the phrase structure would also be straight-forward; e.g., if the last word in

[2]The EM algorithm uses repeated passes through the input data to calculate the *expected value* of the PCFG rule statistics. In principle these statistics can be calculated from any sufficiently large sample from the input language, so in a data-rich environment a learner might simply calculate these expectations from fresh samples, rather than repeatedly reparsing the same input data.

[3]However, note that the EM algorithm only approximately maximizes likelihood: in general EM can "get stuck" at a local optimum of the likelihood which may not be a global optimum, in much the same way that non-statistical learners can (Gibson and Wexler, 1994).

the sentences are verbs, then the language is verb-final. However, without any advance knowledge of the syntactic categories of lexical items or the phrase structure, it seems that a learner would face a "chicken and egg" problem with this grammar and input data. As we will see, ML estimators like EM avoid this problem because they automatically perform *joint inference* for both the phrase structure rules and lexical entries together.

The EM algorithm is an iterative algorithm, so it must be given an "initial guess" for the probabilities of the PCFG rules. Here we initialize the rule probabilities uniformly, e.g., because there are 4 rules expanding VP, each is initially assigned probability 0.25. (We also added a small amount of randomly-varying jitter to the initial rule probabilities for reasons explained below).

The EM algorithm repeatedly parses the sentences in its input data, and uses information gathered from those parses to re-estimate the probabilities of the PCFG rules. Figure 3 displays the values of some of the rule probabilities at each EM iteration. The differences in the probabilities of the high probability rules are not that important (they differ by less than a factor of 10). Rather, the important result is that the high probability rules are more probable than the low probability rules by many orders of magnitude. After six iterations the algorithm has assigned extremely low probabilities to rules that are inconsistent with the input data, such as VP→ NP V and N→*gives*, i.e., a small set of rules capable of generating the strings in the input data have been learned. Thus in this example the EM algorithm is capable of learning both the head-complement ordering inside VP as well as the categories of the corresponding lexical items. (If the input strings are changed so that the verbs follow, rather than precede, their complements, then the EM algorithm assigns low probabilities to the head-initial VP rules instead).

Note that these rule probabilities have been learned *without any direct negative evidence*. Maximum Likelihood estimators can be viewed as indirectly exploiting implicit negative evidence because in general they select grammars which assigns as little probability as possible to strings not in the input data, as this enables the model to increase the probability of the strings that do occur in the input data. None of this is specific to PCFGs: see Clark (2004) for a more detailed discussion of the role of negative evidence in statistical learning.

Because probabilities are required to sum to one, it's possible to assign much higher probabilities — and hence maximize likelihood — if the grammar generates a small set of strings than if the grammar generates a larger set of strings. For example, if probabilities are assigned equally to 10 strings then each will have probability 0.1, while if the grammar generates 100 strings and assigns probability uniformly, then each will have probability 0.01. Now consider the grammar rules in Figure 2. If only the V-initial VP rules have high probability, then *bites* and *gives* are analyzed as Vs, and the grammar generates a relatively small set of strings such as *a dog gives a cat* besides the strings in the input data. However, if the V-final VP rules have high probability then *bites* and *gives* must be analyzed as either Det or Ns and *cat*, *dog* and *bone* must be analyzed as both V and either Det or N. Such a grammar would generate a much larger set of strings (including examples such as *the gives a bites cat*), and hence would necessarily assign them a lower probability.[4]

The learned grammar generates strings beyond those contained in the input data: it generates strings such as *a bone bites a dog* with high probability, even though this string does not appear in the input data. It does this because the grammar's parameters are the rule probabilities, and the values that make the input data most likely also make such unseen strings more likely as well. Thus the PCFG rules and the PCFG "context-free" independence assumptions determine how a ML estimator generalises from the input data. If we were to relax the PCFG independence assumptions by associating a probability parameter with each tree that the grammar generates then a ML estimate

[4]To keep things simple I am assuming here that the grammar only generates a finite set of strings and assigns equal probability to the strings it generates. Using Information Theory this argument can be generalized to cases where the grammar generates infinitely many strings and assigns non-uniform probabilities to them (Cover and Thomas, 2006).

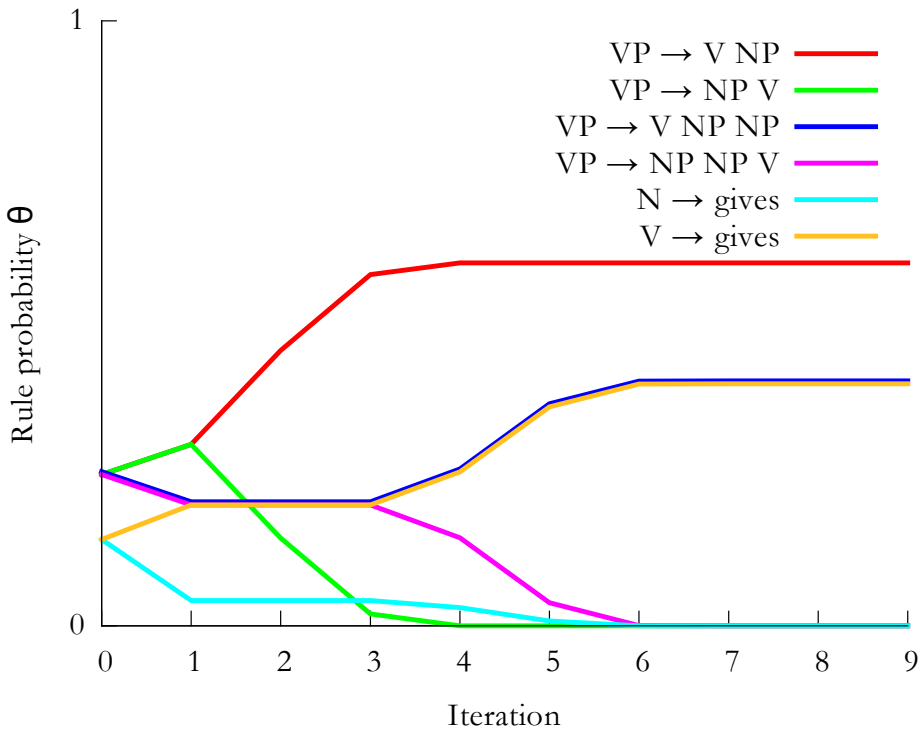

Figure 3: The probabilities of the PCFG rules at each iteration of the Expectation-Maximization algorithm. Notice that after six iterations the algorithm assigns extremely low probabilities to rules that are inconsistent with the input data, such as VP→ NP V and N→ *gives*.

would put positive probability only on the sentences of the input data, and would not generalise to strings not seen in the input data at all. Thus the structure of the model, in particular the details of its independence assumptions, are central to statistical learning, since they are what force the learner to generalize beyond the examples in its input data.

It's interesting to inspect the rule probabilities at early iterations of the EM algorithm. Although the algorithm starts with a uniform assignment of rule probabilities, the probabilities of both the phrasal rules (e.g., VP → V NP) and lexical rules (e.g., V → *gives*) vary simultaneously during the learning process. The EM algorithm avoids the "chicken and egg" problem mentioned earlier because it performs joint inference for both the phrasal and lexical rules, rather than inferring them separately. Learning the lexicon and syntactic structure jointly (i.e., together) seems to be essential here: it's hard to see how any "staged" learner (which attempted to learn lexical entries before learning syntax, or vice versa) could succeed on this data.

It's also easy to see that there are cases where the ML estimators will fail to learn anything meaningful. The PCFG in Figure 2 has rules generating N and Det in either order, and the rules generating lexical items are completely ambiguous. For reasons just explained the ML estimate assigns high probability to only one of the rules NP → Det N and NP → N Det, but which rule "wins" is determined by which has even minutely higher probability in the initial grammar (i.e., the jitter in the initial grammar determines which rules have high probability). Thus the MLE fails to learn a single grammar in this case because the data simply does not contain sufficient information to unambiguously identify the ordering of constituents within NP.

4 Bayesian inference and universal markedness preferences

Bayesian inference extends frequentist methods like Maximum Likelihood estimation by including prior information. Bayesians use Bayes Rule to factorise the *posterior distribution* over parameter values Θ (here the PCFG rule probabilities) given the data as proportional to the product of the *likelihood of the parameters* and a *prior distribution* over the parameters:

$$\underbrace{\mathrm{P}(\Theta \mid \mathrm{Data})}_{\text{Posterior}} \propto \underbrace{\mathrm{P}(\mathrm{Data} \mid \Theta)}_{\text{Likelihood}} \underbrace{\mathrm{P}(\Theta)}_{\text{Prior}}$$

The likelihood here is the same as used in Maximum Likelihood estimation, so Bayesian estimators tend to learn parameter values that concentrate mass on a small set of strings that includes the input data, just like a ML estimator. Bayesian estimation differs from Maximum Likelihood estimation primarily in that the likelihood is multiplied by another term called the *prior*. The prior constitutes information known to the learner in the absence of data, in the same way that Universal Grammar encodes information available to the learner in the absence of any data. However, a Bayesian prior is probabilistic, so as well as hard constraints it can encode soft markedness preferences towards or away from particular parameter settings.

Because the posterior is a product of a likelihood and a prior, in general a parameter vector Θ will have a low posterior probability if either it has low prior probability, i.e., it is inconsistent with the expectations of Universal Grammar, or if it has low likelihood, i.e., it is inconsistent with the data. Thus Bayesian inference can be regarded as search for parameter values that satisfy the prior (i.e., Universal Grammar) and are also consistent with the available input data.

Bayesian estimation is also intimately related to the *Minimum Description Length* (MDL) approach to learning (Rissanen, 1989), where the goal is to find as compact a description of the input data as possible. Specifically, it's possible to show that the log likelihood is the length of the shortest description of the input data under an encoding specified by the prior (MacKay, 2003), so the most compact description in an MDL approach is the model with highest posterior probability in a Bayesian approach.

In the rest of this section we will demonstrate how a Bayesian prior can be used to resolve the learning ambiguity in Maximum Likelihood estimation described at the end of the previous section.

Mathematically speaking, a Bayesian prior can contain arbitrary information about the values of the parameters Θ to be estimated, so one could resolve the learning ambiguity by using the prior to prefer a specific parameter setting, e.g., where the rule NP $\rightarrow$ Det N has a higher probability than the rule NP $\rightarrow$ N Det. Such a prior can be regarded as encoding into Universal Grammar a preference for a Det N ordering within NP.

While it may or may not be linguistically reasonable to assume Universal Grammar contains preferences for particular constituent orders, Bayesian priors can be far more subtle than this. For example, there are Bayesian priors (called "sparse Dirichlet priors") that prefer parameter vectors where most the probabilities of the rules expanding specific categories are close to zero. Such priors can be used to encode a markedness preference that the number of lexical entries for functional categories such as determiners is smaller than the number of lexical entries for lexical categories such as nouns. Note that the prior doesn't specify which words are more likely to be determiners (although there are priors that can do this too); it just says that grammars in which there are fewer determiners are to be preferred by the learner.

Bayesian inference is often computationally more complex than Maximum Likelihood estimation, in part because Bayesian inference estimates a posterior *distribution* over parameters, rather than the point estimates of parameters that ML estimation produces. In the case of inference for PCFG rule probabilities from strings, however, there is an algorithm for approximate Bayesian

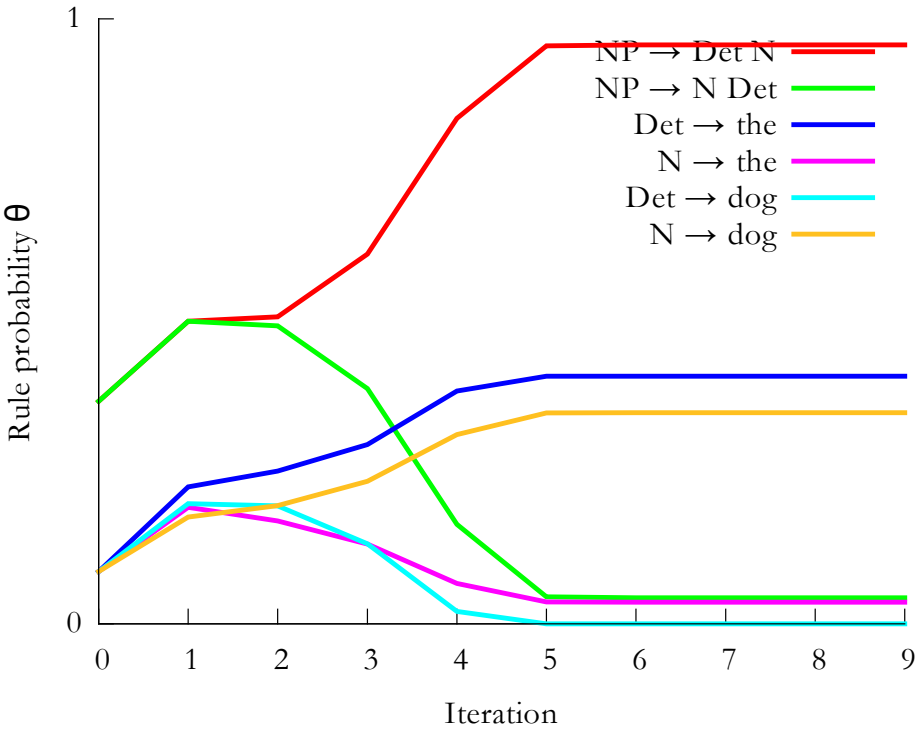

Figure 4: The probabilities of the PCFG rules at each iteration of the Variational Bayes algorithm. Notice that after six iterations the algorithm has identified the NP → Det N is the appropriate rule for expanding NP given this data. The probabilities of the other rules in the grammar follow approximately the trajectories depicted in Figure 3.

inference called *Variational Bayes* (VB) that is a straight-forward generalization of the Expectation Maximization used above (Kurihara and Sato, 2006).

Figure 4 shows the result of Bayesian inference using the VB algorithm when such a sparsity-preferring prior is imposed on the rule probabilities in the Det category and a uniform or "flat" prior is imposed on all other categories. The VB estimator with this prior always prefers the NP → Det N analysis because this enables it to assign a smaller set of words (i.e., *the* and *a*) to the Det category. Thus in this case a Bayesian prior about the relative sparsity of the Det and N categories supplies sufficient information to enable a learner to break the symmetries in the NP category mentioned above.

The reader will note that the VB algorithm assigns non-neglible probabilities to many of the "incorrect" rules (e.g., NP → N Det and N → *the*) in Figure 4, in contrast to the miniscule probabilities assigned to such rules by the ML estimator in Figure 3. This is arguably "a feature, not a bug" of Bayesian inference. Because Bayesian approaches model the entire posterior distribution they more accurately capture uncertainty in the learned models. The input data in the example above only contains 5 sentences, so it's reasonable that a learner might not want to make extremely strong inferences just on the basis of this data. The Bayesian estimates here can be viewed as the learner "keeping its options open" about e.g., whether determiners can be final inside NP. (The inferred probability of the Det → *dog* rule is very low precisely because we imposed a sparsity-preferring prior on the rules expanding Det).

5 More realistic models of language acquisition

The previous two sections described how Maximum Likelihood and Variational Bayes estimators can be used to infer or learn the probabilities of PCFG rules given strings of words as input data. We saw that because ML estimators choose the rule parameters that make the data as likely as possible, they also choose these parameters in a way that assigns as little probability to strings outside the input data as possible. Bayesian estimators combine the likelihood with a prior, which can bias the learner towards certain grammars and away from others. We saw that a sparsity-inducing prior can be used to prefer certain category assignment patterns without requiring prior knowledge of the category of any specific words.

These models are highly simplified in many ways, and it's interesting to investigate whether these methods work in more realistic situations. An obvious experiment is to apply these algorithms to larger grammars and input data. The Expectation Maximization (EM) and Variational Bayes (VB) algorithms are quite efficient, so they can be run with moderate-sized PCFGs on corpora of millions of words. This has been tried with EM (VB is newer and so less explored) and the results are generally disappointing: the inferred grammars tend to generate extremely unnatural phrase-structure analyses, often much worse than simply assigning a uniformly right-branching phrase structure (Klein and Manning, 2004). (An important lesson to draw from this is that while it's relatively easy to come up with learning procedures that do well on toy problems like the example above, it's much harder to develop approaches that work on larger, more realistic input data). While it's still unclear exactly why these EM experiments fail to learn reasonable linguistic analyses, there are at least two reasonable hypotheses: (i) the grammar classes being used are incomplete or otherwise wrong, and (ii) human learners are exploiting more information than is contained in just the strings alone, and there's just not enough information in strings alone to learn a language.

In fact, it's relatively easy to show that the likelihood optimised by the EM algorithm simply does not correctly identify linguistically plausible phrase structure analyses. One way to do this uses a hand-constructed "treebank" of phrase-structure trees, from which the (presumably) correct PCFG rules can be read off and the rule probabilities estimated (recall that ML estimation is easy when the trees are visible). If we run the EM algorithm on the strings of the treebank data and initialize it with the rules and probabilities extracted from the treebank trees, we find that the rule probabilities change dramatically. (The likelihood increases dramatically during this process, so the EM algorithm is in fact functioning as advertised). However, the accuracy of the phrase structure parses generated using this grammar drops dramatically, showing that maximizing likelihood given this input data does not identify rule probabilities that result in the most accurate phrase-structure parses of this data.

Today most work on grammar induction in computational linguistics uses *dependency grammars* rather than PCFGs (Klein and Manning, 2004). These dependency grammars directly generate dependencies between words (i.e., head-to-head dependencies), so a dependency grammar representation seems simpler than a corresponding constituency representation. Recent formal results also suggest that dependency grammars are identifiable from strings alone while arbitrary PCFGs are not (Hsu et al., 2012); this may partly explain why dependency grammars are learned more successfully. However these dependency representations correspond 1-to-1 to certain X' tree representations (Johnson, 2007), so perhaps the conclusion that should be drawn from the failure to infer PCFGs that generate English phrase structure from large corpora is that it's necessary to use a PCFG whose rules are suitably constrained.

Now it's well-known that simple CFGs of the kind used above are grossly inadequate for describing the syntactic structures of natural languages. A reasonable hypothesis is that the way PCFGs associate probabilities with phrase structure rules simply is not an appropriate way of pa-

rameterizing natural language grammars. As noted above, the locality requirements of CFGs don't correspond well with those of natural languages. The Generalized Phrase Structure Grammar (GPSG) for English proposed by Gazdar et al. (1985), while weakly equivalent to a CFG, makes extensive use of "meta-rules" and feature-passing conventions to capture linguistic generalizations, and it's reasonable to expect that devices of similar complexity would be required in any adequate PCFG for English (let alone one that encompasses a reasonable portion of Universal Grammar). So perhaps the reason why EM fails to learn linguistically reasonable PCFGs from large corpora is simply because the PCFGs we've used can't capture the relevant linguistic generalizations. There has not been much work on feature-passing versions of PCFGs analogous to the GPSG extension of CFGs, but this might be one way of developing linguistically more realistic PCFGs: see Goodman (1998) for one attempt.

Given the linguistic problems with PCFGs, it's reasonable to ask whether it's possible to develop probabilistic versions of other kinds of grammars that will enable ML and Bayesian inference to be used with them as well. Abney (1997) showed that *log-linear grammars*, also known as *Maximum Entropy models*, can be used to develop probabilistic versions of virtually any kind of grammar, including Principles and Parameters grammars and Minimalist Grammars. It turns out that these probabilistic grammars strictly generalize PCFGs (i.e., every PCFG is a log-linear model, but there are log-linear grammars that generate non-context-free languages). Log-linear models are very closely related to Harmonic Grammars, which are in turn a stochastic variant of Optimality Grammars (Smolensky and Legendre, 2005).

Log-linear grammars relax the strict locality requirements of PCFGs, allowing dependencies between arbitrary elements in a linguistic structure. This enables them to directly capture a wide range of different linguistic constraints and generalizations, such as WH-movement, raising, etc. This extra expressive power comes at a massive increase in computational complexity, which in general means that estimation of log-linear grammars require additional approximations to be used on top of those used with PCFGs. ML estimation of log-linear grammars requires the computation of a "partition function" and/or its derivatives, which involves summing over the possible structures that the grammar generates (of which there are usually infinitely many) (Johnson and Riezler, 2002).

PCFGs are computationally simpler than log-linear grammars because their strictly local "context-free" dependencies guarantee that their partition function is a constant, so such intractable sums are not required. Interestingly, the relevant notion of context-freeness here applies to derivational structure, rather than the derived surface strings or trees. There are grammar formalisms such as Tree-Adjoining Grammar (TAG) with a context-free derivation structure that can generate non-context-free languages; these grammar formalisms have easily computable partition functions and therefore have many of the attractive computational properties of PCFGs (Resnik, 1992).

While it's straight-forward to formulate a version of the EM algorithm that learns the weights of a log-linear model from strings alone, the computational complexity of such an algorithm is generally assumed to be so high that it would infeasible for realistic grammars and input data. However, the efficient approximation of partition functions and their derivatives is currently the focus of intense research in the field of machine learning (Dahl et al., 2012), and it's not implausible that successes there will lead to more efficient and accurate EM algorithms for log-linear grammars as well.

We now move to the second point: that language learning may require additional information beyond that contained in a set of strings of surface forms. Perhaps language learners need, or can exploit, some kinds of non-linguistic input in order to learn language?

A Bayesian prior can be regarded as providing a learner with information in addition to the input data, in the sense that the prior contains information available to the learner in the absence of experience. Bayesian methods are still comparatively unexplored, but results obtained so far

suggest that an appropriate prior can produce significant but not ground-breaking improvements in language learning compared to ML. For example, (Naseem et al., 2010) uses Bayesian methods and constraints to disambiguate ambiguities in dependency grammars (e.g., should nouns or determiners be taken to be the heads of NPs?), which improves the accuracy of the dependencies the learner identifies.

The subfield of computational linguistics known as *semantic parsing* is concerned with learning the relationship between surface strings and "meaning representations", usually mediated by extremely simple grammars (much of this work focuses on learning a lexicon). For example, Jones et al. (2012) models the relationship between meaning representations and surface strings using tree transducers, while Kwiatkowski et al. (2012) learn a Combinatory Categorial Grammar that maps logical forms (in the lambda calculus) to surface strings. This approach is attractive because the models do not aim to predict the probability of sentences in general, but rather the conditional probability of words and constructions given the meaning the speaker intends to express. Focusing more specifically on the acquisition of a lexicon, Johnson et al. (2012) describes models that exploit *social cues* (e.g., the location of the speaker's gaze) to help infer the intended referent of an utterance, which is presumably helpful for learning word meanings.

Because log-linear grammars are so general, they can be used to model the relationship between surface structures and "meaning representations", but they face the computational challenges mentioned earlier. One standard approach for avoiding the computational intractability of the partition function in log-linear grammars is to use *Conditional Maximum Likelihood* (CML) estimates in place of Maximum Likelihood (ML) estimates. CML optimizes a conditional probability distribution, which requires us to factor the input data items into "input" and "output" representations. (CML cannot be used to directly estimate a grammar from input data consisting of strings alone, as was done in the PCFG examples above, because this data does not come in the form of input/output pairs). In natural language processing parsing applications, where the goal is to recover a parse tree given an input string, it is standard to use CML to estimate the conditional distribution over parse trees conditional on surface strings, i.e., the "input" representations are surface strings and the "output" representations are parse trees. This application of CML produces very accurate parsers for engineering applications (Charniak and Johnson, 2005).

Johnson and Riezler (2002) suggest that it might be linguistically more natural to use CML to estimate conditional models in the opposite direction, i.e., where the "input" representations are underlying forms of some kind (e.g., logical forms or some kind of semantic representation) and the "output" representations are surface strings. This approach views the goal of language acquisition to be a mapping from meaning to form. It does not require learning a distribution over possible underlying forms or meanings, which arguably is not a property of language at all, but depends on the communicative intentions of the language users. In such an approach it would be natural to impose a Bayesian prior that prefers sparse distributions over "outputs" (surface representations) given any "input" (semantic form); this would be a Bayesian formulation of the *uniqueness principle* (Wexler and Culicover, 1980) or the *principle of contrast* (Clark, 1987). It's not unreasonable to hope that these conditional distributions might be close to deterministic, in which case statistical estimation might be used to learn a grammar which is essentially non-stochastic. Of course, because the input data for such a learner consists of underlying form/surface string pairs, the learner must obtain them somehow, and it's not clear how a human learner could obtain the underlying forms required (but see Hsu and Griffiths (2009)).[5] It's worth noting that using CML to estimate log-linear models of the conditional distribution of surface forms given underlying forms is now common in computational phonology (Goldwater and Johnson, 2003; Hayes and Wilson, 2008).

[5]Non-statistical models of language acquisition such as Wexler and Culicover (1980) also assume that the learner's input consist of underlying form/surface form pairs.

Finally, we turn to the topic of learning the constructions or rules of a grammar as well as their probabilities. This requires a significant extension of the statistical estimation techniques described above. The learners described so far are *parametric estimators* because they learn the values of a fixed-dimensional vector of parameters (e.g., rule probabilities) that is specified before learning begins. Learners which learn which parameters from an infinite set of possible parameters are appropriate based on the input data are called *non-parametric estimators*. For example, a learner which learns a set of PCFG rules as well as their probabilities from the input data would be a non-parametric estimator.

A standard approach for learning rules could be described as employing a "let a hundred flowers bloom" strategy. This strategy starts by generating an initial (perhaps randomly chosen) set of rules, and then applies a parametric learner to infer their probabilities from the input data. Then each rules whose probability falls below some threshold is discarded and new rules are proposed somehow, usually on the basis of sentences in the input data that the current grammar parses "badly". Examples of inference procedures employing this strategy include Carroll and Charniak (1992) and Stolcke and Omohundro (1994). A weakness of this strategy is that its performance depends crucially on the procedure for proposing new rules, and these tend to be ad hoc. Such approaches are often viewed as searching for a set of rules and associated probabilities that score well on some Bayesian or MDL metric, but there are usually no guarantees that the algorithms employed will find or even approach an optimum of such a metric.

An obvious application of non-parametric estimation is in the *acquisition of the lexicon*. There's no obvious bound on the number of possible lexical entries in a natural language; rather, the number of lexical entries as well as the information they contain must be learned from the input data. Elman (1990) introduced the *word segmentation* problem (segmenting phonemic representations of utterances into words) as a simplified version of lexical acquisition and Goldwater et al. (2009) developed *Bayesian non-parametric models* for this problem. Bayesian non-parametric approaches have several advantages over "hundred flowers" strategies: they are less ad hoc because they explicitly define the goal of inference (the posterior distribution), and there are provably correct inference algorithms that simultaneously search for a finite set of parameters (i.e., lexical entries) as well as their values (i.e., the probability of each entry) given the input data. Johnson (2008) describes a Bayesian non-parametric framework called "Adaptor Grammars" in which a CFG generates an infinite set of highly structured possible parameters (e.g., lexical entries with morphological category and syllable structure constraints), from which the learner chooses a finite set on the basis of the input data.

6 Conclusion

This paper has discussed a variety of ways in which language acquisition might be understood as statistical inference. We began with a simple example of ML inference of PCFG rule probabilities from a small corpus of strings, and saw how it could learn both lexical entries and phrase structure rules in the absence of negative evidence. We also saw how a Bayesian prior could be used to bias a grammar toward a linguistically reasonable analysis in situations where the input data does not contain sufficient information. Thus on this simple example at least, statistical learners are capable of exploiting distributional information in order to infer linguistic generalisations about their input data.

We then noted that even though these PCFG estimators can be applied to larger, more realistic input data, the results are uniformly disappointing, and discussed two possible reasons: (i) the grammars aren't capturing appropriate linguistic generalizations, and (ii) surface strings on their own don't supply sufficient information to learn language. The search for more linguistically-expressive

grammars lead us to consider log-linear grammars, which are far more expressive than PCFGs but whose estimation can involve major computational challenges. We also discussed approaches that assumed richer input data (say, surface strings/underlying form pairs), and pointed out that it may be both computationally more feasible and linguistically more reasonable to view language acquisition as learning the conditional distribution of surface forms given underlying semantic forms. Finally, we discussed non-parametric inference, which aims to learn which parameters are relevant to the input data, as well as the values that they take.

A central theme of this paper is that because exact calculations are computationally intractable in all but the simplest cases, most statistical learning algorithms involve approximations of one kind or another, and in general we don't understand what kinds of inaccuracies (if any) these approximations induce, and what the linguistic implications of these inaccuracies are. This means that most work in the field adopts an experimental, "try it and see" approach. Since all these models involve approximations and idealisations of one kind or another, it is hard to draw definitive conclusions from them about human language acquisition. When an experiment "fails" it is often difficult to understand why, and even when an experiment "succeeds", all one can say is that it seems that the input data coupled with the model's assumptions suffice to learn the phenomena in question: even then it's possible that success is due to idealisations about the input data or algorithmic approximations unlikely to generalise to more realistic scenarios.

It's probably fair to say that most of the research discussed above comes from the computer science literature, and focuses on algorithmic details and implementations. Perhaps this is not surprising, given that there are major technical challenges involved in developing inference procedures capable of learning human languages, and there are still major gaps in our understanding of the theoretical and practical application of these inference algorithms that such technical expertise will be required to resolve. But I suspect that ultimately the most important insights in this work will turn out to be linguistic rather than computational. For example, it's already the case that the log-linear approach to probabilistic grammars can capture much richer and more complex linguistic generalizations than are currently being explored, and in principle it should be possible for Bayesian priors to express the kinds of rich linguistic knowledge that linguists posit for Universal Grammar. It would be extremely interesting to investigate just what a statistical estimator using linguistically plausible parameters might be able to learn.

Acknowledgements

Avery Andrews, Benjamin Börschinger, Alexander Clark, Stephen Crain, Katherine Demuth and Mark Steedman all provided insightful comments on early drafts; naturally all responsibility for errors remain my own. This research was supported by Australian Reseach Council Discovery Projects DP110102506 and DP110102593.

References

Abney, S. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4):597–617.

Carroll, G. and E. Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Proceedings of the AAAI Workshop on Statistically-Based Natural Language Processing Techniques*, San Jose, CA.

Charniak, E. and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative

reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 173–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Chomsky, N. 1957. *Syntactic Structures*. Mouton, The Hague.

Chomsky, N. 1986. *Knowledge of Language: Its Nature, Origin and Use*. Praeger, New York.

Clark, A. 2004. Grammatical inference and first language acquisition. In *Psychocomputational Models of Human Language Acquisition*, pages 25–32, Geneva.

Clark, E. 1987. The principle of contrast: A constraint on language acquisition. In B. MacWhinney, editor, *Mechanisms of Language Acquisition*, pages 1–33. Lawrence Erlbaum, New Jersey.

Cohen, S. and N. Smith. 2012. Empirical risk minimization for probabilistic grammars: Sample complexity and hardness of learning. *Computational Linguistics*, 38(3):479–526.

Cover, T. and J. Thomas. 2006. *Elements of Information Theory*. Wiley Series in Telecommunications and Signal Processing. John Wiley and Sons.

Dahl, G., R. Adams, and H. Larochelle. 2012. Training restricted Boltzmann machines on word observations. In J. Langford and J. Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 679–686, New York, NY, USA. Omnipress.

Elman, J. 1990. Finding structure in time. *Cognitive Science*, 14:197–211.

Fodor, J. and S. Crain. 1987. Simplicity and generality of rules in language acquisition. In B. MacWhinney, editor, *Mechanisms of Language Acquisition*, pages 35–65. Lawrence Erlbaum, New Jersey.

Gazdar, G., E. Klein, G. Pullum, and I. Sag. 1985. *Generalized Phrase Structure Grammar*. Basil Blackwell, Oxford.

Geman, S. and M. Johnson. 2004. Probability and statistics in computational linguistics, a brief review. In M. Johnson, S. P. Khudanpur, M. Ostendorf, and R. Rosenfeld, editors, *Mathematical Foundations of Speech and Language Processing*, pages 1–26. Springer, New York.

Gibson, E. and K. Wexler. 1994. Triggers. *Linguistic Inquiry*, 25(3):407–454.

Goldwater, S., T. L. Griffiths, and M. Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Goldwater, S. and M. Johnson. 2003. Learning OT constraint rankings using a Maximum Entropy model. In J. Spenader, A. Eriksson, and O. Dahl, editors, *Proceedings of the Stockholm Workshop on Variation within Optimality Theory*, pages 111–120, Stockholm. Stockholm University.

Goodman, J. 1998. *Parsing inside-out*. PhD thesis, Harvard University.

Hayes, B. and C. Wilson. 2008. A Maximum Entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39(3):379–440.

Headden III, W. P., M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado. Association for Computational Linguistics.

Hsu, A. and T. Griffiths. 2009. Differential use of implicit negative evidence in generative and discriminative language learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 754–762, 2009.

Hsu, D., S. Kakade, and P. Liang. 2012. Identifiability and unmixing of latent parse trees. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1520–1528. MIT Press, Cambridge, MA.

Johnson, M. 2007. Transforming projective bilexical dependency grammars into efficiently-parsable CFGs with Unfold-Fold. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 168–175, Prague, Czech Republic. Association for Computational Linguistics.

Johnson, M. 2008. Using Adaptor Grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, pages 398–406, Columbus, Ohio. Association for Computational Linguistics.

Johnson, M., K. Demuth, and M. Frank. 2012. Exploiting social information in grounded language learning via grammatical reduction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 883–891, Jeju Island, Korea. Association for Computational Linguistics.

Johnson, M. and S. Riezler. 2002. Statistical models of syntax learning and use. *Cognitive Science*, 26: 239–253.

Jones, B., M. Johnson, and S. Goldwater. 2012. Semantic parsing with Bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 488–496, Jeju Island, Korea. Association for Computational Linguistics.

Klein, D. and C. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 478–485, 2004.

Kurihara, K. and T. Sato. 2006. Variational Bayesian grammar induction for natural language. In *8th International Colloquium on Grammatical Inference*, 2006.

Kwiatkowski, T., S. Goldwater, L. Zettlemoyer, and M. Steedman. 2012. A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 234–244, Avignon, France. Association for Computational Linguistics.

Lari, K. and S. Young. 1990. The estimation of Stochastic Context-Free Grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4(35-56).

MacKay, D. J. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Marr, D. 1982. *Vision*. W.H. Freeman and Company, New York.

Naseem, T., H. Chen, R. Barzilay, and M. Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244, Cambridge, MA. Association for Computational Linguistics.

Neal, R. M. and G. E. Hinton. 2001. A new view of the EM algorithm that justifies incremental and other variants. In M. I. Jordan and T. J. Sejnowski, editors, *Graphical Models: Foundations of Neural Computation*. The MIT Press, Cambridge, Massachusetts.

Nocedal, J. and S. J. Wright. 2006. *Numerical Optimization*. Springer, New York.

Pereira, F. and Y. Schabes. 1992. Inside-Outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, San Francisco. Morgan Kaufmann.

Resnik, P. 1992. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of the 14th conference on Computational linguistics - Volume 2*, COLING '92, pages 418–424, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rissanen. 1989. *Stochastic Complexity in Statistical Inquiry*. World Scientific Company, Singapore.

Shieber, S. M. 1985. Evidence against the Context-Freeness of natural language. *Linguistics and Philosophy*, 8(3):333–344.

Smolensky, P. and G. Legendre. 2005. *The Harmonic Mind: From Neural Computation To Optimality-Theoretic Grammar*. The MIT Press.

Stolcke, A. and S. Omohundro. 1994. Inducing probabilistic grammars by Bayesian model merging. In R. C. Carrasco and J. Oncina, editors, *Grammatical Inference and Applications*, pages 106–118. Springer, New York.

Wexler, K. and P. Culicover. 1980. *Formal Principles of Language Acquisition*. The MIT Press, Cambridge, Massachusetts.