PROBABILITY AND STATISTICS IN COMPUTATIONAL LINGUISTICS, A BRIEF REVIEW

STUART GEMAN* AND MARK JOHNSON*

1. Introduction. Computational linguistics studies the computational processes involved in language learning, production, and comprehension. Computational linguists believe that the essence of these processes (in humans and machines) is a computational manipulation of information. Computational *psycholinguistics* studies psychological aspects of human language (e.g., the time course of sentence comprehension) in terms of such computational processes.

Natural language processing is the use of computers for processing natural language text or speech. Machine translation (the automatic translation of text or speech from one language to another) began with the very earliest computers [Kay et al., 1994]. Natural language interfaces permit computers to interact with humans using natural language, e.g., to query databases. Coupled with speech recognition and speech synthesis, these capabilities will become more important with the growing popularity of portable computers that lack keyboards and large display screens. Other applications include spell and grammar checking and document summarization. Applications outside of natural language include compilers, which translate source code into lower-level machine code, and computer vision [Foo, 1974, Foo, 1982].

The notion of a grammar is central to most work in computational linguistics and natural language processing. A *grammar* is a description of a language; usually it identifies the sentences of the language and provides descriptions of them, e.g., by defining the phrases of a sentence, their inter-relationships, and perhaps also aspects of their meanings. *Parsing* is the process of recovering a sentence's description from its words, while *generation* is the process of translating a meaning or some other part of a sentence's description into a *grammatical* or well-formed sentence. Parsing and generation are major research topics in their own right. Evidently, human use of language involves some kind of parsing and generation process, as do many natural language processing applications. For example, a machine translation program may parse an input language sentence into a (partial) representation of its meaning, and then generate an output language sentence from that representation.

Although the intellectual roots of modern linguistics go back thousands of years, by the 1950s there was considerable interest in applying the then newly developing ideas about finite-state machines and other kinds of automata, both deterministic and stochastic, to natural language. Automata

*Department of Cognitive and Linguistic Sciences, Brown University, Providence, RI 02912, USA.

are Markov-like machines consisting of a set of states and a set of allowed state-to-state transitions. An input sequence, selected from a finite input alphabet, moves the machine from state to state along allowed transitions. [Chomsky, 1957] pointed out clearly the inadequacies of finite-state machines for modelling English syntax. An effect of Chomsky's observations, perhaps unintended, was to discourage further research into probabilistic and statistical methods in linguistics. In particular, stochastic grammars were largely ignored. Instead, there was a shift away from simple automata, both deterministic and stochastic, towards more complex non-stochastic grammars, most notably "transformational" grammars. These grammars involved two levels of analyses, a "deep structure" meant to capture more-or-less simply the meaning of a sentence, and a "surface structure" which reflects the actual way in which the sentence was constructed. The deep structure might be a clause in the active voice, "Sandy saw Sam," whereas the surface structure might involve the more complex passive voice, "Sam was seen by Sandy."

Transformational grammars are computationally complex, and in the 1980s several linguists came to the conclusion that much simpler kinds of grammars could describe most syntactic phenomena, developing Generalized Phrase-Structure Grammars [Gazdar et al., 1985] and Unification-based Grammars [Kaplan and Bresnan, 1982, Pollard and Sag, 1987], [Shieber, 1986]. These grammars generate surface structures directly; there is no separate deep structure and therefore no transformations. These kinds of grammars can provide very detailed syntactic and semantic analyses of sentences, but as explained below, even today there are no comprehensive grammars of this kind that fully accommodate English or any other natural language.

Natural language processing using hand-crafted non-stochastic grammars suffers from two major drawbacks. First, the syntactic coverage offered by any available grammar is incomplete, reflecting both our lack of understanding of even relatively frequently occuring syntactic constructions and the organizational difficulty of manually constructing any artifact as complex as a grammar of a natural language. Second, such grammars almost always permit a large number of *spurious ambiguities*, i.e., parses which are permitted by the rules of syntax but have unusual or unlikely semantic interpretations. For example, in the sentence *I saw the boat with the telescope*, the prepositional phrase *with the telescope* is most easily interpreted as the instrument used in seeing, while in *I saw the policeman with the rifle*, the prepositional phrase usually receives a different interpretation in which the policeman has the rifle. Note that the corresponding alternative interpretation is marginally accessible for each of these sentences: in the first sentence one can imagine that the telescope is on the boat, and in the second, that the rifle (say, with a viewing scope) was used to view the policeman.

In effect, there is a dilemma of coverage. A grammar rich enough to accommodate natural language, including rare and sometimes even "ungrammatical" constructions, fails to distinguish natural from unnatural interpretations. But a grammar sufficiently restricted so as to exclude what is unnatural fails to accommodate the scope of real language. These observations lead, in the 1980's, to a renewed interest in stochastic approaches to natural language, particularly to speech. Stochastic finite-state automata became the basis of speech recognition systems by out-performing the best of the systems based on deterministic hand-crafted grammars. Largely inspired by the success of stochastic approaches in speech recognition, computational linguists began applying them to other natural language processing applications. Usually, the architecture of such a stochastic model is specified manually (e.g., the possible states of a stochastic finite-state automaton and the allowed transitions between them), while the model's parameters are estimated from a *training corpus*, i.e., a large representative sample of sentences.

As explained in the body of this paper, stochastic approaches replace the binary distinctions (grammatical versus ungrammatical) of non-stochastic approaches with probability distributions. This provides a way of dealing with the two drawbacks of non-stochastic approaches. Ill-formed alternatives can be characterized as extremely low probability rather than ruled out as impossible, so even ungrammatical strings can be provided with an interpretation. Similarly, a stochastic model of possible interpretations of a sentence provides a method for distinguishing more plausible interpretations from less plausible one.

The next section, §2, introduces formally various classes of grammars and languages. Probabilistic grammars are introduced in §3, along with the basic issues of parametric representation, inference, and computation.

2. Grammars and languages. The formal framework, whether used in a transformational grammar, a generalized phrase-structure grammar, or a more traditionally styled context-free grammar, is due to [Chomsky, 1957] and his co-workers. In this section, we will present a brief introduction to this framework. But for a thorough (and very readable) presentation we highly recommend the book by [Hopcroft and Ullman, 1979].

If T is a finite set of symbols, let T^* be the set of all strings (i.e., finite sequences) of symbols of T, including the empty string, and let T^+ be the set of all nonempty strings of symbols of T. A *language* is a subset of T^*. A *rewrite grammar* G is a quadruple $G = (T, N, S, R)$, where T and N are disjoint finite sets of symbols (called the *terminal* and *non-terminal* symbols respectively), $S \in N$ is a distinguished non-terminal called the *start symbol*, and R is a finite set of productions. A *production* is a pair (α, β) where $\alpha \in N^+$ and $\beta \in (N \cup T)^*$; productions are usually written $\alpha \to \beta$. Productions of the form $\alpha \to \epsilon$, where ϵ is the empty string, are called *epsilon productions*. In this paper we will restrict attention to

grammars without epsilon productions, i.e., $\beta \in (N \cup T)^+$, as this simplifies the mathematics considerably.

A rewrite grammar G defines a *rewriting relation* $\Rightarrow_G \subseteq (N \cup T)^* \times (N \cup T)^*$ over pairs of strings consisting of terminals and nonterminals as follows: $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$ iff $\alpha \rightarrow \beta \in R$ and $\gamma, \delta \in (N \cup T)^*$ (the subscript G is dropped when clear from the context). The reflexive, transitive closure of $\Rightarrow$ is denoted $\Rightarrow^*$. Thus $\Rightarrow^*$ is the rewriting relation using arbitrary finite sequences of productions. (It is called "reflexive" because the identity rewrite, $\alpha \Rightarrow^* \alpha$, is included). The *language generated by* G, denoted L_G, is the set of all strings $w \in T^+$ such that $S \Rightarrow^* w$.

A terminal or nonterminal $X \in N \cup T$ is *useless* unless there are $\gamma, \delta \in (N \cup T)^*$ and $w \in T^*$ such that $S \Rightarrow^* \gamma X \delta \Rightarrow^* w$. A production $\alpha \rightarrow \beta \in R$ is useless unless there are $\gamma, \delta \in (N \cup T)^*$ and $w \in T^*$ such that $S \Rightarrow^* \gamma\alpha\delta \Rightarrow \gamma\beta\delta \Rightarrow^* w$. Informally, useless symbols or productions never appear in any sequence of productions rewriting the start symbol S to any sequence of terminal symbols, and the language generated by a grammar is not affected if useless symbols and productions are deleted from the grammar.

Example 1. Let the grammar $G_1 = (T_1, N_1, \mathrm{S}, R_1)$, where $T_1 = \{\text{grows}, \text{rice}, \text{wheat}\}$, $N_1 = \{\mathrm{S}, \mathrm{NP}, \mathrm{VP}\}$ and $R_1 = \{\mathrm{S} \rightarrow \mathrm{NP\,VP}, \mathrm{NP} \rightarrow \text{rice}, \mathrm{NP} \rightarrow \text{wheat}, \mathrm{VP} \rightarrow \text{grows}\}$. Informally, the nonterminal S rewrites to sentences or clauses, NP rewrites to noun phrases and VP rewrites to verb phrases. Then $L_{G_1} = \{\text{rice grows}, \text{ wheat grows}\}$. G_1 does not contain any useless symbols or productions.

Rewrite grammars are traditionally classified by the shapes of their productions. $G = (T, N, S, R)$ is a *context-sensitive grammar* iff for all productions $\alpha \rightarrow \beta \in R$, $|\alpha| \leq |\beta|$, i.e., the right-hand side of each production is not shorter than its left-hand side. G is a *context-free grammar* iff $|\alpha| = 1$, i.e., the left-hand side of each production consists of a single non-terminal. G is a *left-linear grammar* iff G is context-free and β (the right-hand side of the production) is either of the form Aw or of the form w where $A \in N$ and $w \in T^*$; in a *right-linear grammar* β always is of the form wA or w. A right or left-linear grammar is called a *regular* grammar.

It is straight-forward to show that the classes of languages generated by these classes of grammars stand in equality or subset relationships. Specifically, the class of languages generated by right-linear grammars is the same as the class generated by left-linear grammars; this class is called the *regular languages*, and is a strict subset of the class of languages generated by context-free grammars, which is a strict subset of the class of languages generated by context-sensitive grammars, which in turn is a strict subset of the class of languages generated by rewrite grammars.

The computational complexity of deciding whether a string is generated by a rewrite grammar is determined by the class that the grammar belongs to. Specifically, the *recognition problem for grammar* G takes a string $w \in T^+$ as input and returns TRUE iff $w \in L_G$. Let $\mathcal{G}$ be a class of

grammars. The *universal recognition problem for* $\mathcal{G}$ takes as input a string $w \in T^+$ and a grammar $G \in \mathcal{G}$ and returns TRUE iff $w \in L_G$.

There are rewriting grammars that generate languages that are *recursively enumerable* but not *recursive*. In essence, a language L_G is recursively enumerable if there exists an algorithm that is guaranteed to halt and emit TRUE whenever $w \in L_G$, and *may* halt and emit NOT TRUE, *or may not halt*, whenever $w \notin L_G$. On the other hand, a language is recursive if there exists an algorithm that always halts, and emits TRUE or NOT TRUE depending on whether $w \in L_G$ or $w \notin L_G$, respectively.[1] Obviously, the set of recursive languages is a subset of the set of recursively enumerable languages. The recognition problem for a language that is recursively enumerable but not recursive is said to be *undecidable*. Since such languages do exist, generated by rewrite grammars, the universal recognition problem for rewrite grammars is undecidable.

The universal recognition problem for context-sensitive grammars *is* decidable, and furthermore is in PSPACE (space polynomial in the size of G and w), but there are context-sensitive grammars for which the recognition problem is PSPACE-complete [Garey and Johnson, 1979], so the universal recognition problem for context-sensitive grammars is PSPACE-complete also. Since NP$\subseteq$PSPACE, we should not expect to find a polynomial-time recognition algorithm for arbitrary context-sensitive grammars. The universal recognition problem for context-free grammars is decidable in time polynomial in the size of w and linear in the size of G; as far as we are aware a tight upper bound is not known. Finally, the universal recognition problem for regular grammars is decidable in time linear in w and G.

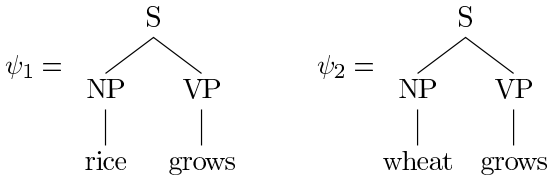
It turns out that context-sensitive grammars (where a production rewrites more than one nonterminal) have not had many applications in natural language processing, so from here on we will concentrate on context-free grammars, where all productions take the form $A \rightarrow \beta$, where $A \in N$ and $\beta \in (N \cup T)^+$.

An appealing property of grammars with productions in this form is that they induce tree structures on the strings that they generate. And, as we shall see shortly (§3), this is the basis for bringing in probability distributions and the theory of inference. We say that the context-free grammar $G = (T, N, S, R)$ generates the labelled, ordered tree ψ iff the root node of ψ is labelled S, and for each node n in ψ, either n has no children and its label is a member of T (i.e., it is labelled with a terminal) or else there is a production $A \rightarrow \beta \in R$ where the label of n is A and the left-to-right sequence of labels of n's immediate children is β. It is straight forward to show that w is in L_G iff G generates a tree ψ whose *yield* (i.e., the left-to-right sequence of terminal symbols labelling ψ's leaf nodes) is

[1] A rigorous definition requires a proper introduction to Turing machines. Again, we recommend [Hopcroft and Ullman, 1979].

w; ψ is called a *parse tree* of w (with respect to G). In what follows, we define Ψ_G to be the set of parse trees generated by G, and $\mathcal{Y}(\cdot)$ to be the function that maps trees to their yields.

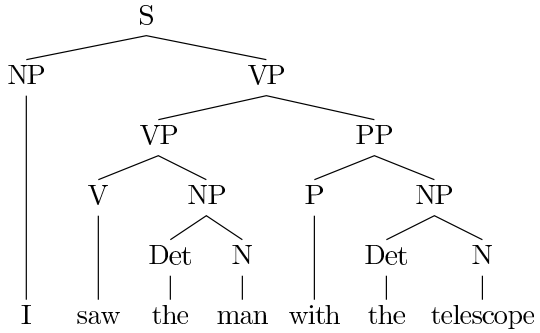
Example 1 (continued). The grammar G_1 defined above generates the following two trees, ψ_1 and ψ_2.

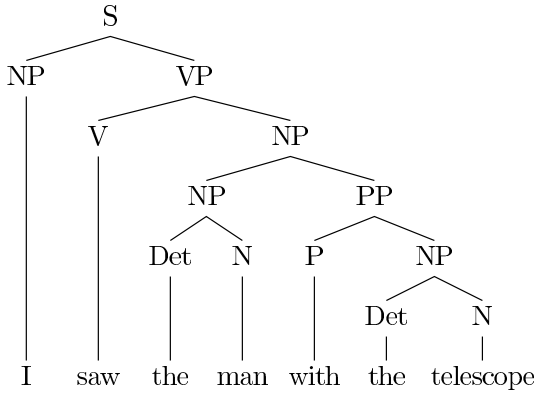


In this example, $\mathcal{Y}(\psi_1)$ = *rice grows* and $\mathcal{Y}(\psi_2)$ = *wheat grows*

A string of terminals w is called *ambiguous* iff w has two or more parse trees. Linguistically, each parse tree of an ambiguous string usually corresponds to a distinct interpretation.

Example 2. Consider $G_2 = (T_2, N_2, \mathrm{S}, R_2)$, where $T_2 = \{\mathrm{I}, \mathrm{saw}, \mathrm{the}, \mathrm{man}, \mathrm{with}, \mathrm{telescope}\}$, $N_2 = \{\mathrm{S}, \mathrm{NP}, \mathrm{N}, \mathrm{Det}, \mathrm{VP}, \mathrm{V}, \mathrm{PP}, \mathrm{P}\}$ and $R_2 = \{\mathrm{S} \to \mathrm{NP\,VP}, \mathrm{NP} \to \mathrm{I}, \mathrm{NP} \to \mathrm{Det\,N}, \mathrm{Det} \to \mathrm{the}, \mathrm{NP} \to \mathrm{NP\,PP}, \mathrm{N} \to \mathrm{man}, \mathrm{N} \to \mathrm{telescope}, \mathrm{VP} \to \mathrm{V\,NP}, \mathrm{VP} \to \mathrm{VP\,PP}, \mathrm{PP} \to \mathrm{P\,NP}, \mathrm{V} \to \mathrm{saw}, \mathrm{P} \to \mathrm{with}\}$. Informally, N rewrites to nouns, Det to determiners, V to verbs, P to prepositions and PP to prepositional phrases. It is easy to check that the two trees ψ_3 and ψ_4 with the yields $\mathcal{Y}(\psi_3) = \mathcal{Y}(\psi_4) =$ *I saw the man with the telescope* are both generated by G_2. Linguistically, these two parse trees represent two different syntactic analyses of the sentence. The first analysis corresponds to the interpretation where the seeing is by means of a telescope, while the second corresponds to the interpretation where the man has a telescope.



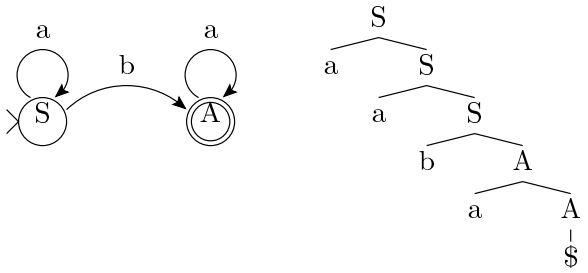


There is a close relationship between linear grammars and finite-state machines. A finite-state machine is a kind of automaton that makes state-to-state transitions driven by letters from an input alphabet (see [Hopcroft and Ullman, 1979]) for details). Each finite-state machine has a corresponding right-linear grammar which has the property that the set of strings accepted by the machine is the same as the set of strings generated by the grammar (modulo an endmarker, as discussed below), and the nonterminals of this grammar are exactly the set of states of the machine. Moreover, there is an isomorphism between accepting computations of the machine and parse trees generated by this grammar: for each sequence of states that the machine transitions through in an accepting computation there is a parse tree of the corresponding grammar containing exactly the same sequence of states (the example below clarifies this).

The grammar $G_M = (T, N, S, R)$ that corresponds to a finite-state machine M is one where the nonterminal symbols N are the states of M, the start symbol S is the start state of M, and the terminal symbols T are the input symbols to M together with a new symbol '\$', called the *endmarker*, that does not appear in the transition labels of M. The productions R of G_M come in two kinds. R can contain productions $A \rightarrow b\,B$, where $A, B \in N$ and $b \in T$, iff there is a transition in M from state A to state B on input symbol b. R contains the production $A \rightarrow \$$ iff A is a final state in M. Informally, M accepts a string $w \in T^*$ iff w is the sequence of inputs along a path from M's start state to some final state. It is easy to show that G_M generates the string $w\$$ iff M accepts the string w. (If we permitted epsilon productions then it would not be necessary to use an endmarker; R would contain a production $A \rightarrow \epsilon$, where ϵ is the empty string, iff A is a final state in M).

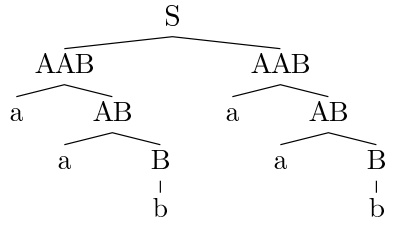
Example 3. Consider the right-linear grammar $G_3 = (T_3, N_3, \mathrm{S}, R_3)$, where $T_3 = \{\mathrm{a}, \mathrm{b}, \$\}$, $N_3 = \{\mathrm{S}, \mathrm{A}\}$ and $R_3 = \{\mathrm{S} \rightarrow \mathrm{b}, \mathrm{S} \rightarrow \mathrm{a\,S}, \mathrm{S} \rightarrow \mathrm{b\,A}, \mathrm{A} \rightarrow \mathrm{a\,A}, \mathrm{A} \rightarrow \$\}$. G_3 corresponds to the finite-state machine depicted below, where the '>' attached to the state labelled S indicates it is the

start state, and the double circle indicates that the state labelled A is a final state. The parse tree for 'aaba$' with respect to G_3 translates immediately into the sequence of states that the machine transitions through when accepting 'aaba'.



As remarked earlier, context-sensitive and unrestricted rewrite grammars do not seem to be useful in many natural language processing applications. On the other hand, the notation of context-free grammars is not ideally suited to formulating natural language grammars. Furthermore, it is possible to show that some natural languages are not context-free languages [Culy, 1985, Shieber, 1985]. These two factors have led to the development of a variety of different kinds of grammars. Many of these can be described as *annotated phrase structure grammars*, which are extensions of context-free grammars in which the set of nonterminals N is very large, possibly infinite, and N and R possess a linguistically motivated structure. In Generalized Phrase Structure Grammars [Gazdar et al., 1985] N is finite, so these grammars always generate context-free languages, but in unification grammars such as Lexical-Functional Grammar [Kaplan and Bresnan, 1982] or Head-driven Phrase Structure Grammar [Pollard and Sag, 1987] N is infinite and the languages such grammars generate need not be context-free or even recursive.

Example 4. Let $G_4 = (T_4, N_4, \mathrm{S}, R_4)$ where $T_4 = \{\mathrm{a}, \mathrm{b}\}$, $N_4 = \{\mathrm{S}\} \cup \{\mathrm{A}, \mathrm{B}\}^+$ (i.e., N_4 consists of S and nonempty strings over the alphabet A, B) and $R_4 = \{\mathrm{S} \to \alpha\,\alpha : \alpha \in \{\mathrm{A}, \mathrm{B}\}^+\} \cup \{\mathrm{A} \to \mathrm{a}, \mathrm{B} \to \mathrm{b}\} \cup \{\mathrm{A}\alpha \to \mathrm{a}\,\alpha, \mathrm{B}\alpha \to \mathrm{b}\,\alpha : \alpha \in \{\mathrm{A}, \mathrm{B}\}^+\}$. G_4 generates the language $\{ww : w \in \{\mathrm{a}, \mathrm{b}\}^+\}$, which is not a context-free language. A parse tree for aabaab is shown below.



3. Probability and statistics. Obviously broad coverage is desirable—natural language is rich and diverse, and not easily held to a small set of rules. But it is hard to achieve broad coverage without massive ambiguity (a sentence may have tens of thousands of parses), and this of course complicates applications like language interpretation, language translation, and speech recognition. This is the dilemma of coverage that we referred to earlier, and it sets up a compelling role for probabilistic and statistical methods.

We will review the main probabilistic grammars and their associated theories of inference. We begin in §3.1 with probabilistic regular grammars, also known as hidden Markov models (HMM), which are the foundation of modern speech recognition systems. In §3.2 we discuss probabilistic context-free grammars, which turn out to be essentially the same thing as branching processes. We review the estimation problem, the computation problem, and the role of criticality. Finally, in §3.3, we take a more general approach to placing probabilities on grammars, which leads to Gibbs distributions, a role for Besag's *pseudolikelihood method* [Besag, 1974, Besag, 1975], various computational issues, and, all in all, an active area of research in computational linguistics.

3.1. Hidden Markov models and regular grammars. Recall that a right-linear grammar $G = (T, N, S, R)$ corresponding to a finite-state machine is characterized by rewrite rules of the form $A \to b\,B$ or $A \to \$$, where $A, B \in N$, $b \in T$, and $\$ \in T$ is a special terminal that we call an endmarker. The connection with Hidden Markov Models (HMM's) is transparent: N defines the states, R defines the allowed transitions (A can go to B if there exists a production of the form $A \to b\,B$), and the string of terminals defines the "observation." The process is "hidden" since, in general, the observations do not uniquely define the sequence of states.

In general, it is convenient to work with a "normal form" for right-linear grammars: all rules are either of the form $A \to b\,B$ or $A \to b$, where $A, B \in N$ and $b \in T$. It is easy to show that every right-linear grammar has an equivalent normal form in the sense that the two grammars produce the same language. Essentially nothing is lost, and we will usually work with a normal form.

3.1.1. Probabilities. Assume that R has no useless symbols or productions. Then the grammar G can be made into a probabilistic grammar by assigning to each nonterminal $A \in N$ a probability distribution p over productions of the form $A \to \alpha \in R$: for every $A \in N$

$$\sum_{\substack{\alpha \in (N \cup T)^+ \\ \text{s.t. } (A \to \alpha) \in R}} p(A \to \alpha) = 1 \ . \tag{1}$$

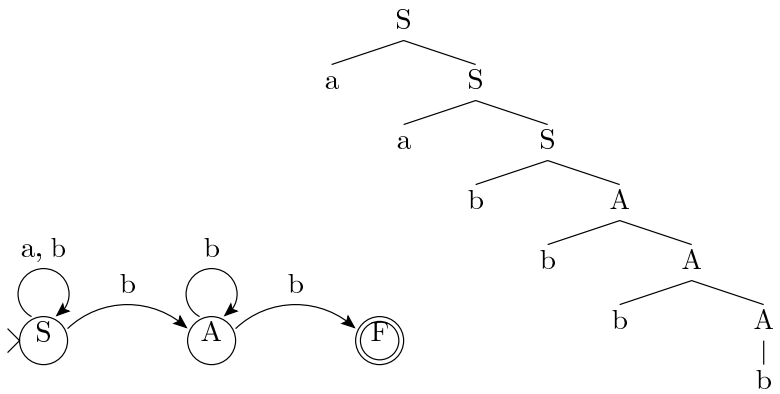
Recall that Ψ_G is the set of parse trees generated by G (see §2). If G is linear, then $\psi \in \Psi_G$ is characterized by a sequence of productions, starting

from S. It is, then, straightforward to use p to define a probability P on Ψ_G: just take $P(\psi)$ (for $\psi \in \Psi_G$) to be the product of the associated production probabilities.

Example 5. Consider the right-linear grammar $G_5 = (T_5, N_5, \mathrm{S}, R_5)$, with $T_5 = \{\mathrm{a}, \mathrm{b}\}$, $N_5 = \{\mathrm{S}, \mathrm{A}\}$ and the productions (R_5) and production probabilities (p):

$$\begin{array}{lcll} \mathrm{S} & \rightarrow & \mathrm{a}\,\mathrm{S} & p = .80 \\ \mathrm{S} & \rightarrow & \mathrm{b}\,\mathrm{S} & p = .01 \\ \mathrm{S} & \rightarrow & \mathrm{b}\,\mathrm{A} & p = .19 \\ \mathrm{A} & \rightarrow & \mathrm{b}\,\mathrm{A} & p = .90 \\ \mathrm{A} & \rightarrow & \mathrm{b} & p = .10\ . \end{array}$$

The language is the set of strings ending with a sequence of at least two *b*'s. The grammar is ambiguous: in general, a sequence of terminal states does not uniquely identify a sequence of productions. The sentence *aabbbb* has three parses (determined by the placement of the production $\mathrm{S} \rightarrow \mathrm{b}\,\mathrm{A}$), but the most likely parse, by far, is $\mathrm{S} \rightarrow \mathrm{a}\,\mathrm{S}$, $\mathrm{S} \rightarrow \mathrm{a}\,\mathrm{S}$, $\mathrm{S} \rightarrow \mathrm{b}\,\mathrm{A}$, $\mathrm{A} \rightarrow \mathrm{b}\,\mathrm{A}$, $\mathrm{A} \rightarrow \mathrm{b}\,\mathrm{A}$, $\mathrm{A} \rightarrow \mathrm{b}$ ($P = .8 \cdot .8 \cdot .19 \cdot .9 \cdot .1$), which has a *posterior* probability of nearly .99. The corresponding parse tree is shown below.



An equivalent formulation is through the associated three-state (S, A, and F) two-output (a and b) HMM also shown above: the transition probability matrix is

$$\begin{pmatrix} .81 & .19 & .00 \\ .00 & .90 & .10 \\ .00 & .00 & 1.00 \end{pmatrix}$$

where the first row and column represent S, the next represent A, and the last represent F; and the output probabilities are based on state-to-state pairs,

$$(\mathrm{S},\mathrm{S}) \to \begin{cases} a & \text{prob} = 80/81 \\ b & \text{prob} = 1/81 \end{cases}$$

$$(\mathrm{S},\mathrm{A}) \to \begin{cases} a & \text{prob} = 0 \\ b & \text{prob} = 1 \end{cases}$$

$$(\mathrm{A},\mathrm{F}) \to \begin{cases} a & \text{prob} = 0 \\ b & \text{prob} = 1 \ . \end{cases}$$

3.1.2. Inference. The problem is to estimate the transition probabilities, $p(\cdot)$, either from parsed data (examples from Ψ_G) or just from sentences (examples from L_G). Consider first the case of parsed data ("supervised learning"), and let $\psi_1, \psi_2, \ldots, \psi_n \in \Psi$ be a sequence taken iid according to P. If $f(A \to \alpha; \psi)$ is the counting function, counting the number of times the transition $A \to \alpha \in R$ occurs in ψ, then the likelihood function is

$$L = L(p; \psi_1, \ldots, \psi_n) = \prod_{i=1}^{n} \prod_{A \to \alpha \in R} p(A \to \alpha)^{f(A \to \alpha; \psi_i)} \ . \tag{2}$$

The maximum likelihood estimate is, sensibly, the relative frequency estimator:

$$\hat{p}(A \to \alpha) = \frac{\sum_{i=1}^{n} f(A \to \alpha; \psi_i)}{\sum_{i=1}^{n} \sum_{\beta \text{ s.t. } A \to \beta \in R} f(A \to \beta; \psi_i)} \ . \tag{3}$$

If a nonterminal A does not appear in the sample, then the numerator and denominator are zero, and $\hat{p}(A \to \alpha)$, $\alpha \in (N \cup T)^+$, can be assigned arbitrarily, provided it is consistent with (1).

The problem of estimating p from sentences ("unsupervised learning") is more interesting, and more important for applications. Recall that $\mathcal{Y}(\psi)$ is the yield of ψ, i.e. the sequence of terminals in ψ. Given a sentence $w \in T^+$, let Ψ_w be the set of parses which yield w: $\Psi_w = \{\psi \in \Psi : \mathcal{Y}(\psi) = w\}$. The likelihood of a sentence $w \in T^+$ is the sum of the likelihoods of its possible parses:

$$L(p; w) = \sum_{\psi \in \Psi_w} P(\psi) = \sum_{\psi \in \Psi_w} \prod_{A \to \alpha \in R} p(A \to \alpha)^{f(A \to \alpha; \psi)} \ .$$

Imagine now a sequence $\psi_1, \ldots, \psi_n$, iid according to P, for which only the corresponding yields, $w_i = \mathcal{Y}(\psi_i)$, $1 \le i \le n$, are observed. The likelihood function is

$$L = L(p; w_1, \ldots, w_n) = \prod_{i=1}^{n} \sum_{\psi \in \Psi_{w_i}} \prod_{A \to \alpha \in R} p(A \to \alpha)^{f(A \to \alpha; \psi_i)} \ . \tag{4}$$

To get the maximum likelihood equation, take logarithms, introduce Lagrange multipliers to enforce (1), and set the derivative with respect to $p(A \to \alpha)$ to zero:

$$\lambda_A + \frac{1}{\hat{p}(A \to \alpha)} \sum_{i=1}^{n} \frac{\sum_{\psi \in \Psi_{w_i}} f(A \to \alpha; \psi) \prod_{B \to \beta \in R} \hat{p}(B \to \beta)^{f(B \to \beta; \psi)}}{\sum_{\psi \in \Psi_{w_i}} \prod_{B \to \beta \in R} \hat{p}(B \to \beta)^{f(B \to \beta; \psi)}} = 0 \,. \tag{5}$$

Introduce $E_p[\cdot]$, meaning expectation under the probability on P induced by p, and solve for $\hat{p}(A \to \alpha)$:

$$\hat{p}(A \to \alpha) = \frac{\sum_{i=1}^{n} E_{\hat{p}}[f(A \to \alpha; \psi) | \psi \in \Psi_{w_i}]}{\sum_{A \to \beta \in R} \sum_{i=1}^{n} E_{\hat{p}}[f(A \to \beta; \psi) | \psi \in \Psi_{w_i}]} \,. \tag{6}$$

We can't solve, directly, for $\hat{p}$, but (6) suggests an iterative approach [Baum, 1972]: start with an arbitrary $\hat{p}_0$ (but positive on R). Given $\hat{p}_t$, $t = 1, 2, \ldots$, define $\hat{p}_{t+1}$ by using $\hat{p}_t$ in the right hand side of (6):

$$\hat{p}_{t+1}(A \to \alpha) = \frac{\sum_{i=1}^{n} E_{\hat{p}_t}[f(A \to \alpha; \psi) | \psi \in \Psi_{w_i}]}{\sum_{A \to \beta \in R} \sum_{i=1}^{n} E_{\hat{p}_t}[f(A \to \beta; \psi) | \psi \in \Psi_{w_i}]} \,. \tag{7}$$

Evidently, $\hat{p}_t = \hat{p}_{t+1}$ if and only if we have found a solution to the likelihood equation, $\frac{\partial}{\partial \hat{p}(A \to \alpha)} L = 0$, $\forall A \to \alpha \in R$. What's more, as shown by [Baum, 1972], it turns out that $L(\hat{p}_{t+1}; w_1, \ldots, w_n) \geq L(\hat{p}_t; w_1, \ldots, w_n)$, and the procedure finds a local maximum of the likelihood. It turns out, as well, that (7) is just an instance of the EM algorithm, which of course is more general and was discovered later by [Dempster et al., 1977].

Needless to say, nothing can be done with this unless we can actually evaluate, in a computationally feasible way, expressions like $E_{\hat{p}}[f(A \to \alpha; \psi) | \psi \in \Psi_w]$. This is one of several closely related computational problems that are part of the mechanics of working with grammars.

3.1.3. Computation. A sentence $w \in T^+$ is *parsed* by finding a sequence of productions $A \to b\,B \in R$ which yield w. Depending on the grammar, this corresponds more or less to an *interpretation* of w. Often, there are many parses and we say that w is ambiguous. In such cases, if there is a probability p on R then there is a probability P on Ψ, and a reasonably compelling choice of parse is the most likely parse:

$$\arg \max_{\psi \in \Psi_w} P(\psi) \,. \tag{8}$$

This is the *maximum a posteriori* (MAP) estimate of ψ—obviously it minimizes the probability of error under the distribution P.

What is the probability of w? How are its parses computed? How is the most likely parse computed? These computational issues turn out to be

more-or-less the same as the issue of computing $E_{\hat{p}}[f(A \to \alpha; \psi)|\psi \in \Psi_w]$ that came up in our discussion of inference. The basic structure and cost of the computational algorithm is the same for each of the four problems—compute the probability of w, compute the set of parses, compute the best parse, compute $E_{\hat{p}}$. For regular grammars, there is a simple dynamic programming solution to each of these problems, and in each case the complexity is of the order $n \cdot |R|$, where n is the length of w, and $|R|$ is the number of productions in G.

Consider the representative problem of producing the most likely parse, (8). Let $w = (b_1, \ldots, b_n) \in T^n$. There are $n-1$ productions of the form $A_k \to b_{k+1}\, A_{k+1}$ for $k = 0, \ldots, n-2$, with $A_0 = S$, followed by a single terminating production $A_{n-1} \to b_n$. The most likely sequence of productions can be computed by a dynamic-programming type iteration: for every $A \in N$ initialize with

$$\begin{aligned} \mathcal{A}_1(A) &= S \\ \mathcal{V}_1(A) &= p(S \to b_1 A)\ . \end{aligned}$$

Then, given $\mathcal{A}_k(A)$ and $\mathcal{V}_k(A)$, for $A \in N$ and $k = 1, 2, \ldots, n-2$, compute $\mathcal{A}_{k+1}(A)$ and $\mathcal{V}_{k+1}(A)$ from

$$\begin{aligned} \mathcal{A}_{k+1}(A) &= \arg\max_{B \in N} p(B \to b_{k+1} A)\mathcal{V}_k(B) \\ \mathcal{V}_{k+1}(A) &= p(\mathcal{A}_{k+1}(A) \to b_{k+1} A)\mathcal{V}_k(\mathcal{A}_{k+1}(A))\ . \end{aligned}$$

Finally, let

$$\mathcal{A}_n = \arg\max_{B \in N} p(B \to b_n)\mathcal{V}_{n-1}(B).$$

Consider the most likely sequence of productions from S at "time 0" to A at "time k," given $b_1, \ldots, b_k$, $k = 1, \ldots, n-1$. $\mathcal{A}_k(A)$ is the state at time $k-1$ along this sequence, and $\mathcal{V}_k(A)$ is the likelihood of this sequence. Therefore, $\hat{A}_{n-1} \stackrel{def}{=} \mathcal{A}_n$ is the state at time $n-1$ associated with the most likely parse, and working backwards, the best state sequence overall is $\hat{A}_0, \hat{A}_1, \ldots, \hat{A}_{n-1}$, where

$$\hat{A}_{k-1} = \mathcal{A}_k(\hat{A}_k), \quad k = n-1, n-2, \ldots, 1\ .$$

There can be ties when more than one sequence achieves the optimum. In fact, the procedure generalizes easily to produce the best l parses, for any $l > 1$. Another modification produces all parses, while still another computes expectations E_p of the kind that appear in the EM iteration (7) or probabilities such as $P\{\mathcal{Y}(\psi) = w\}$ (these last two are, essentially, just a matter of replacing argmax by summation).

3.1.4. Speech recognition. An outstanding application of probabilistic regular grammars is to speech recognition. The approach was first proposed in the 1970's (see [Jelinek, 1997] for a survey), and has since become the dominant technology. Modern systems achieve high accuracy in multi-user continuous-speech applications. Many tricks of representation and computation are behind the successful systems, but the basic technology is nevertheless that of probabilistic regular grammars trained via EM and equipped with a dynamic programming computational engine. We will say something here, briefly and informally, about how these systems are crafted.

So far in our examples T has generally represented a vocabulary of words, but it is not words themselves that are observable in a speech recognition task. Instead, the acoustic signal is observable, and a time-localized discrete representation of this signal makes up the vocabulary T. A typical approach is to start with a spectral representation of progressive, overlapping windows, and to summarize this representation in terms of a relatively small number, perhaps 200, of possible values for each window. One way to do this is with a clustering method such as vector quantization. This ensemble of values then constitutes the terminal set T.

The state space, N, and the transition rules, R, are built from a hierarchy of models, for phonemes (which correspond to letters in speech), words, and grammars. A phoneme model might have, for example, three states representing the beginning, middle, and end of a phoneme's pronunciation, and transitions that allow, for example, remaining in the middle state as a way of modeling variable duration. The state space, N, is small—maybe three states for each of thirty or forty phonemes—making a hundred or so states. This becomes a regular grammar by associating the transitions with elements of T, representing the quantized acoustic features. Of course a realistic system must accommodate an enormous variability in the acoustic signal, even for a single speaker, and this is why probabilities are so important.

Words are modeled similarly, as a set of phonemes with a variety of allowed transition sequences representing a variety of pronunciations choices. These representations can now be expanded into basic units of phoneme pronunciation, by substituting phoneme models for phonemes. Although the transition matrix is conveniently organized by this hierarchical structure, the state space is now quite large: the number of words in the system's vocabulary (say 5,000) *times* the number of states in the phoneme models (say 150). In fact many systems model the effects of context on articulation (e.g. co-articulation), often by introducing states that represent *triplets* of phonemes ("triphones"), which can further increase the size of N, possibly dramatically.

The sequence of words uttered in continuous speech is highly constrained by syntactic and semantic conventions. These further constraints, which amount to a grammar on words, constitute a final level in the hi-

erarchy. An obvious candidate model would be a regular grammar, with N made up of syntactically meaningful parts of speech (verb, noun, noun phrase, article, and so on). But implementations generally rely on the much simpler and less structured *trigram*. The set of states is the set of ordered word pairs, and the transitions are *a priori* only limited by noting that the second word at one unit of time must be the same as the first word at the next. Obviously, the trigram model is of no utility by itself; once again probabilities play an essential role in meaningfully restricting the coverage.

Trigrams have an enormous effective state space, which is made all the larger by expanding the words themselves in terms of word models. Of course the actual number of possible, or at least reasonable, transitions out of a state in the resulting (expanded) grammar is not so large. This fact, together with a host of computational and representational tricks and compromises, renders the dynamic programming computation feasible, so that training can be carried out in a matter of minutes or hours, and recognition can be performed at real time, all on a single user's PC.

3.2. Branching processes and context-free grammars. Despite the successes of regular grammars in speech recognition, the problems of language *understanding* and *translation* are generally better addressed with the more structured and more powerful context-free grammars. Following our development of probabilistic regular grammars in the previous section, we will address here the inter-related issues of fitting context-free grammars with probability distributions, estimating the parameters of these distributions, and computing various functionals of these distributions.

The context-free grammars $G = (T, N, S, R)$ have rules of the form $A \to \alpha, \alpha \in (N \cup T)^+$, as discussed previously in §2. There is again a normal form, known as the Chomsky normal form, which is particularly convenient when developing probabilistic versions. Specifically, one can always find a context-free grammar G', with all productions of the form $A \to BC$ or $A \to a$, $A, B, C, \in N$, $a \in T$, which produces the same language as G: $L_{G'} = L_G$. Henceforth, we will assume that context-free grammars are in the Chomsky normal form.

3.2.1. Probabilities. The goal is to put a probability distribution on the set of parse trees generated by a context-free grammar in Chomsky normal form. Ideally, the distribution will have a convenient parametric form, that allows for efficient inference and computation.

Recall from §2 that context-free grammars generate labeled, ordered trees. Given sets of nonterminals N and terminals T, let Ψ be the set of finite trees with:

(a) root node labeled S;
(b) leaf nodes labeled with elements of T;
(c) interior nodes labeled with elements of N;
(d) every nonterminal (interior) node having either two children labeled with nonterminals or one child labeled with a terminal.

Every $\psi \in \Psi$ defines a sentence $w \in T^+$: read the labels off of the terminal nodes of ψ from left to right. Consistent with the notation of §3.1, we will write $\mathcal{Y}(\psi) = w$. Conversely, every sentence $w \in T^+$ defines a subset of Ψ, which we denote by Ψ_w, consisting of all ψ with yield w ($\mathcal{Y}(\psi) = w$). A context-free grammar G defines a subset of Ψ, Ψ_G, whose collection of yields is the language, L_G, of G. We seek a probability distribution P on Ψ which concentrates on Ψ_G.

The time-honored approach to probabilistic context-free grammars is through the production probabilities $p : R \to [0, 1]$, with

$$\sum_{\substack{\alpha \in N^2 \cup T \\ \text{s.t. } (A \to \alpha) \in R}} p(A \to \alpha) = 1 \ . \tag{9}$$

Following the development in §3.1, we introduce a counting function $f(A \to \alpha; \psi)$, which counts the number of instances of the rule $A \to \alpha$ in the tree ψ, i.e. the number of nonterminal nodes A whose daughter nodes define, left-to-right, the string α. Through f, p induces a probability P on Ψ:

$$P(\psi) = \prod_{(A \to \alpha) \in R} p(A \to \alpha)^{f(A \to \alpha; \psi)} \ . \tag{10}$$

It is clear enough that P concentrates on Ψ_G, and we shall see shortly that this parameterization, in terms of products of probabilities p, is particularly workable and convenient. The pair, G and P, is known as a probabilistic context-free grammar, or PCFG for short.

Branching Processes and Criticality. Notice the connection to branching processes [Harris, 1963]: Starting at S, use R, and the associated probabilities $p(\cdot)$, to expand nodes into daughter nodes until all leaf nodes are labeled with terminals (elements of T). Since branching processes display critical behavior, whereby they may or may not terminate with probability one, we should ask ourselves whether p truly defines a probability on Ψ_G—bearing in mind that Ψ includes only finite trees. Evidently, for p to induce a probability on Ψ_G ($P(\Psi_G) = 1$), the associated branching process must terminate with probability one. This may not happen, as is most simply illustrated by a bare-boned example:

Example 6. $G_6 = (T_6, N_6, \mathrm{S}, R_6)$, $T_6 = \{\mathrm{a}\}$, $N_6 = \{\mathrm{S}\}$, and R_6 includes only

$$\begin{aligned} \mathrm{S} &\to \mathrm{S}\,\mathrm{S} \\ \mathrm{S} &\to \mathrm{a} \ . \end{aligned}$$

Let $p(\mathrm{S} \to \mathrm{S}\,\mathrm{S}) = q$ and $p(\mathrm{S} \to \mathrm{a}) = 1 - q$, and let S_h be the total probability of all trees with depth less than or equal to h. Then $S_2 = 1 - q$ (corresponding to $\mathrm{S} \to \mathrm{a}$) and $S_3 = (1 - q) + q(1 - q)^2$ (corresponding to $\mathrm{S} \to \mathrm{a}$ or $\mathrm{S} \to \mathrm{S}\,\mathrm{S}$ followed by $\mathrm{S} \to \mathrm{a}$, $\mathrm{S} \to \mathrm{a}$). In general, $S_{h+1} = 1 - q + qS_h^2$,

which is nonincreasing in q and converges to $\min(1, \frac{1-q}{q})$ as $q \uparrow 1$. Hence $P(\Psi_G) = 1$ if and only if $q \leq .5$.

More generally, it is not difficult to characterize production probabilities that put full mass on finite trees (so that $P(\Psi_G) = 1$), see for example [Grenander, 1976] or [Harris, 1963]. But the issue is largely irrelevant, since maximum likelihood estimated probabilities *always* have this property, as we shall see shortly.

3.2.2. Inference. As with probabilistic regular grammars, the production probabilities of a context-free grammar, which amount to a parameterization of the distribution P on Ψ_G, can be estimated from examples. In one scenario, we have access to a sequence $\psi_1, \ldots, \psi_n$ from Ψ_G under P. This is "supervised learning," in the sense that sentences come equipped with parses. More interesting is the problem of "unsupervised learning," wherein we observe only the yields, $\mathcal{Y}(\psi_1), \ldots, \mathcal{Y}(\psi_n)$.

In either case, the treatment of maximum likelihood estimation is essentially identical to the treatment for regular grammars. In particular, the likelihood for fully observed data is again (2), and the maximum likelihood estimator is again the relative frequency estimator (3). And, in the unsupervised case, the likelihood is again (4) and this leads to the same EM-type iteration given in (7).

Criticality. We remarked earlier that the issue is largely irrelevant. This is because estimated probabilities $\hat{p}$ are *always* proper probabilities: $\hat{P}(\Psi) = 1$ whenever $\hat{P}$ is induced by $\hat{p}$ computed from (3) or *any* iteration of (7) [Chi and Geman, 1998].

3.2.3. Computation. There are four basic computations: find the probability of a sentence $w \in T^+$, find a $\psi \in \Psi$ (or find *all* $\psi \in \Psi$) satisfying $\mathcal{Y}(\psi) = w$ ("parsing"); find

$$\arg \max_{\substack{\psi \in \Psi \text{ s.t.} \\ \mathcal{Y}(\psi) = w}} P(\psi)$$

("maximum *a posteriori*" or "optimal" parsing); compute expectations of the form $E_{\hat{p}_t}[f(A \to \alpha; \psi) | \psi \in \Psi_w]$ that arise in iterative estimation schemes like (7). The four computations turn out to be more-or-less the same, as was the case for regular grammars (§3.1.3), and there is a common dynamic-programming-like solution [Lari and Young, 1990, Lari and Young, 1991].

We illustrate with the problem of finding the probability of a string (sentence) w, under a grammar G, and under a probability distribution P concentrating on Ψ_G. For PCFGs, the dynamic-programming algorithm involves a recursion over *substrings* of the string w to be parsed. If $w = w_1 \ldots w_m$ is the string to be parsed, then let $w_{i,j} = w_i \ldots w_j$ be the substring consisting of terminals i through j, with the convention that $w_{i,i} = w_i$. The dynamic-programming algorithm works from smaller to

larger substrings $w_{i,j}$, calculating the probability that $A \Rightarrow^\star w_{i,j}$ for each nonterminal $A \in N$. Because a substring of length 1 can only be generated by a unary production of the form $A \rightarrow x$, for each $i = 1, \ldots, m$, $P(A \Rightarrow^\star w_{i,i}) = p(A \rightarrow w_i)$. Now consider a substring $w_{i,j}$ of length 2 or greater. Consider any derivation $A \Rightarrow^\star w_{i,j}$. The first production used must be a binary production of the form $A \rightarrow BC$, with $A, B, C \in N$. That is, there must be a k between i and j such that $B \Rightarrow^* w_{i,k}$ and $C \Rightarrow^* w_{k+1,j}$. Thus the dynamic programming step involves iterating from smaller to larger substrings $w_{i,j}, 1 \leq i, j \leq m$, calculating:

$$P(A \Rightarrow^\star w_{i,j}) = \sum_{\substack{B,C \in N \\ s.t.\ A \rightarrow B\,C \in R}} p(A \rightarrow B\,C) \sum_{k=i,j-1} P(B \Rightarrow^\star w_{i,k}) P(C \Rightarrow^\star w_{k+1,j})\,.$$

At the end of this iteration, $P(w) = P(S \Rightarrow^\star w_{1,m})$. This calculation involves applying each production once for each triple of "string positions" $0 < i \leq k < j \leq m$, so the calculation takes $O(|R|m^3)$ time.

3.3. Gibbs distributions. There are many ways to generalize. The coverage of a context-free grammar may be inadequate, and we may hope, therefore, to find a workable scheme for placing probabilities on context-sensitive grammars, or perhaps even more general grammars. Or, it may be preferable to maintain the structure of a context-free grammar, especially because of its dynamic programming principle, and instead generalize the class of probability distributions away from those induced (parameterized) by production probabilities. But nothing comes for free. Most efforts to generalize run into nearly intractable computational problems when it comes time to parse or to estimate parameters.

Many computational linguists have experimented with using Gibbs distributions, popular in statistical physics, to go beyond production-based probabilities, while nevertheless preserving the basic context-free structure. We shall take a brief look at this particular formulation, in order to illustrate the various challenges that accompany efforts to generalize the more standard probabilistic grammars.

3.3.1. Probabilities. The sample space is the same: Ψ is the set of finite trees, rooted at S, with leaf nodes labeled from elements of T and interior nodes labeled from elements of N. For convenience we will stick to Chomsky normal form, and we can therefore assume that every nonterminal node has either two children labeled from N or a single child labeled from T. Given a particular context-free grammar G, we will be interested in measures concentrating on the subset Ψ_G of Ψ. The sample space, then, is effectively Ψ_G rather than Ψ.

Gibbs measures are built from sums of more-or-less simple functions, known as "potentials" in statistical physics, defined on the sample space. In linguistics, it is more natural to call these *features* rather than potentials. Let us suppose, then, that we have identified M linguistically salient

features $f_1, \ldots, f_M$, where $f_k : \Psi_G \to R$, through which we will characterize the fitness or appropriateness of a structure $\psi \in \Psi_G$. More specifically, we will construct a class of probabilities on Ψ_G which depend on $\psi \in \Psi_G$ only through $f_1(\psi), \ldots, f_M(\psi)$. Examples of features are the number of times a particular production occurs, the number of words in the yield, various measures of subject-verb agreement, and the number of embedded or independent clauses.

Gibbs distributions have the form

$$P_\theta(\psi) = \frac{1}{Z} \exp\left\{ \sum_{i=1}^{M} \theta_i f_i(\psi) \right\} \tag{11}$$

where $\theta_1 \ldots, \theta_M$ are parameters, to be adjusted "by hand" or inferred from data, $\theta = (\theta_1 \ldots, \theta_M)$, and where $Z = Z(\theta)$ (known as the "partition function") normalizes so that $P_\theta(\Psi) = 1$. Evidently, we need to assume or ensure that $\sum_{\psi \in \Psi_G} \exp\{\sum_1^M \theta_i f_i(\psi)\} < \infty$. For instance, we had better require that $\theta_1 < 0$ if $M = 1$ and $f_1(\psi) = |\mathcal{Y}(\psi)|$ (the number of words in a sentence), unless of course $|\Psi_G| < \infty$.

Relation to Probabilistic Context-Free Grammars. The feature set $\{f(A \to \alpha; \psi)\}_{A \to \alpha \in R}$ represents a particularly important special case: The Gibbs distribution (11) takes on the form

$$P_\theta(\psi) = \frac{1}{Z} \exp\left\{ \sum_{A \to \alpha \in R} \theta_{A \to \alpha} f(A \to \alpha; \psi) \right\} . \tag{12}$$

Evidently, we recover probabilistic context-free grammars by taking $\theta_{A \to \alpha} = \log_e p(A \to \alpha)$, where p is a system of production probabilities consistent with (9), in which case $Z = 1$. But is (12) *more general*? Are there probabilities on Ψ_G of this form that are not PCFGs? The answer turns out to be no, as was shown by [Chi, 1999] and [Abney et al., 1999]: Given a probability distribution P on Ψ_G of the form of (12), there always exists a system of production probabilities p under which P is a PCFG.

One interesting consequence relates to the issue of criticality raised in §3.2.1. Recall that a system of production probabilities p may define (through 10) an improper probability P on Ψ_G: $P(\Psi_G) < 1$. In these cases it is tempting to simply renormalize, $\bar{P}(\psi) = p(\psi)/P(\Psi_G)$, but then what kind of distribution is $\bar{P}$? It is clear enough that $\bar{P}$ is Gibbs with feature set $\{f(A \to \alpha; \psi)\}_{A \to \alpha \in R}$, so it must also be a PCFG, by the result of Chi and Abney et al. What are the new production probabilities, $\bar{p}(\cdot)$?

For each $A \in N$, consider the grammar G_A which "starts at A," i.e. replace S, the start symbol, by A. If Ψ_A is the resulting set of tree structures (rooted at A), then (12) defines a measure P_A on Ψ_A, which will have a new normalization Z_A. Consider now the production $A \to B\,C$,

$A, B, C \in N$. Chi's proof of the equivalence between PCFGs and Gibbs distributions of the form (12) is constructive:

$$p(A \to B\,C) = \frac{Z_B Z_C}{Z_A} e^{\theta_{A \to B\,C}}$$

is, explicitly, the production probability under which P is a PCFG. For a terminal production, $A \to a$,

$$p(A \to a) = \frac{1}{Z_A} e^{\theta_{A \to a}} \ .$$

Consider again example 6, in which $S \to S\,S$ with probability q and $S \to a$ with probability $1 - q$. We calculated $P(\Psi_G) = \min(1, \frac{1-q}{q})$, so renormalize and define

$$\bar{P}(\psi) = \frac{P(\psi)}{\min(1, \frac{1-q}{q})} \quad \psi \in \Psi_G \ .$$

Then $\bar{P} = P$ when $q \leq .5$. In any case, $\bar{P}$ is Gibbs of the form (12), with $\theta_{S \to S\,S} = \log_e q$, $\theta_{S \to a} = \log_e (1 - q)$, and $Z_S = \min(1, \frac{1-q}{q})$. Accordingly, $\bar{P}$ is also a PCFG with production probabilities

$$\bar{p}(S \to S\,S) = q \frac{\min(1, \frac{1-q}{q}) \min(1, \frac{1-q}{q})}{\min(1, \frac{1-q}{q})} = q \min(1, \frac{1-q}{q})$$

and

$$\bar{p}(S \to a) = (1 - q) \frac{1}{\min(1, \frac{1-q}{q})} \ .$$

In particular, $\bar{p} = p$ when $q \leq .5$, but $\bar{p}(S \to S\,S) = 1 - q$ and $\bar{p}(S \to a) = q$ when $q > .5$.

3.3.2. Inference. The feature set $\{f_i\}_{i=1,\ldots,M}$ can accommodate arbitrary linguistic attributes and constraints, and the Gibbs model (11), therefore, has great promise as an accurate measure of linguistic fitness. But the model depends critically on the parameters $\{\theta_i\}_{i=1,\ldots,M}$, and the associated estimation problem is, unfortunately, very hard. Indeed, the problem of unsupervised learning appears to be all but intractable.

Of course if the features are simply production frequencies, then (11) is just a PCFG, with $Z = 1$, and the parameters are just log production probabilities ($\theta_{A \to \alpha} = \log_e p(A \to \alpha)$) and easy to estimate (see §3.2.2). More generally, let $\theta = (\theta_1, \ldots, \theta_M)$ and suppose that we observe a sample $\psi_1, \ldots \psi_n \in \Psi_G$ ("supervised learning"). Writing Z as $Z(\theta)$, to emphasize the dependency of the normalizing constant on θ, the likelihood function is

$$L = L(\theta; \psi_1 \ldots, \psi_n) = \prod_{i=1}^{n} \frac{1}{Z(\theta)} \exp\left\{ \sum_{j=1}^{M} \theta_j f_j(\psi_i) \right\}$$

which leads to the likelihood equations (by setting $\frac{\partial}{\partial \theta_j} \log L$ to 0):

$$E_\theta[f_j(\psi)] = \frac{1}{n} \sum_{i=1}^{n} f_j(\psi_i) \qquad 1 \leq j \leq N \tag{13}$$

where E_θ is expectation under (11). In general, Ψ_G is infinite and depending on the features $\{f_i\}_{i=1,\ldots,M}$ and the choice of θ, various sums (like $Z(\theta)$ and E_θ) could diverge and be infinite. But if these summations converge, then the likelihood function is concave. Furthermore, unless there is a linear dependence among $\{f_i\}_{i=1,\ldots,M}$ on $\{\psi_i\}_{i=1,\ldots,n}$, then the likelihood is in fact strictly concave, and there is a unique solution to (13). (If there is a linear dependence, then there are infinitely many θ values with the same likelihood.)

The favorable shape of $L(\theta; \psi_1 \ldots, \psi_n)$ suggests gradient ascent, and in fact the θ_j component of the gradient is proportional to $\frac{1}{n} \sum_{i=1}^{n} f_j(\psi_i) - E_\theta[f_j(\psi)]$. But $E_\theta[f]$ is difficult to compute (to say the least), except in some very special and largely uninteresting cases. Various efforts to use Monte Carlo methods to approximate $E_\theta[f]$, or related quantities that arise in other approaches to estimation, have been made [Abney, 1997]. But realistic grammars involve hundreds or thousands of features and complex feature structures, and under such circumstances Monte Carlo methods are notoriously slow to converge. Needless to say, the important problem of *unsupervised* learning, wherein only yields are seen, is even more daunting.

This state of affairs has prompted a number of suggestions in way of compromise and approximation. One example is the method of pseudolikelihood, which we will now discuss.

Pseudolikelihood. If the primary goal is to select good parses, then perhaps the likelihood function

$$\prod_{i=1}^{n} P_\theta(\psi_i) \tag{14}$$

asks for too much, or even the wrong thing. It might be more relevant to maximize the likelihood of the observed parses, *given the yields* $\mathcal{Y}(\psi_1), \ldots, \mathcal{Y}(\psi_n)$ [Johnson et al., 1999]:

$$\prod_{i=1}^{n} P_\theta(\psi_i | \mathcal{Y}(\psi_i)) \ . \tag{15}$$

One way to compare these criteria is to do some (loose) asymptotics. Let $P(\psi)$ denote the "true" distribution on Ψ_G (from which $\psi_1, \ldots, \psi_n$

are presumably drawn, iid), and in each case (14 and 15) compute the large-sample-size average of the log likelihood:

$$\begin{aligned}
\frac{1}{n}\log\prod_{i=1}^{n}P_\theta(\psi_i) &= \frac{1}{n}\sum_{i=1}^{n}\log P_\theta(\psi_i) \\
&\approx \sum_{\psi\in\Psi_G} P(\psi)\log P_\theta(\psi) \\
&= \sum_{\psi\in\Psi_G} P(\psi)\log P(\psi) - \sum_{\psi\in\Psi_G} P(\psi)\log\frac{P(\psi)}{P_\theta(\psi)} \\
\frac{1}{n}\log\prod_{i=1}^{n}P_\theta(\psi_i|\mathcal{Y}(\psi_i)) &= \frac{1}{n}\sum_{i=1}^{n}\log P_\theta(\psi_i|\mathcal{Y}(\psi_i)) \\
&\approx \sum_{w\in T^+} P(w) \sum_{\substack{\psi\in\Psi_G \\ \text{s.t. } \mathcal{Y}(\psi)=w}} P(\psi|\mathcal{Y}(\psi))\log P_\theta(\psi|\mathcal{Y}(\psi)) \\
&= \sum_{w\in T^+} P(w) \sum_{\substack{\psi\in\Psi_G \\ \text{s.t. } \mathcal{Y}(\psi)=w}} P(\psi|\mathcal{Y}(\psi))\log P(\psi|\mathcal{Y}(\psi)) \\
&\quad - \sum_{w\in T^+} P(w) \sum_{\substack{\psi\in\Psi_G \\ \text{s.t. } \mathcal{Y}(\psi)=w}} P(\psi|\mathcal{Y}(\psi))\log\frac{P(\psi|\mathcal{Y}(\psi))}{P_\theta(\psi|\mathcal{Y}(\psi))} \ .
\end{aligned}$$

Therefore, maximizing the likelihood (14) is more or less equivalent to minimizing the Kullback-Leibler divergence between $P(\psi)$ and $P_\theta(\psi)$, whereas maximizing the "pseudolikelihood" (15) is more or less equivalent to minimizing the Kullback-Leibler divergence between $P(\psi|\mathcal{Y}(\psi))$ and $P_\theta(\psi|\mathcal{Y}(\psi))$—averaged over yields. Perhaps this latter minimization makes more sense, given the goal of producing good parses.

Maximization of (15) is an instance of Besag's remarkably effective *pseudolikelihood method* [Besag, 1974, Besag, 1975], which is commonly used for estimating parameters of Gibbs distributions. The computations involved are generally much easier than what is involved in maximizing the ordinary likelihood function (14). Take a look at the gradient of the logarithm of (15): the θ_j component is proportional to

$$\frac{1}{n}\sum_{i=1}^{n} f_j(\psi_i) - \frac{1}{n}\sum_{i=1}^{n} E_\theta[f_j(\psi)|\mathcal{Y}(\psi)=\mathcal{Y}(\psi_i)] \ . \tag{16}$$

Compare this to the gradient of the likelihood function, which involves $E_\theta[f_j(\psi)]$ instead of $\frac{1}{n}\sum_{i=1}^{n} E_\theta[f_j(\psi)|\mathcal{Y}(\psi)=\mathcal{Y}(\psi_i)]$. $E_\theta[f_j(\psi)]$ is essentially intractable, whereas $E_\theta[f_j(\psi)|\mathcal{Y}(\psi)]$ can be computed directly from the set of parses of the sentence $\mathcal{Y}(\psi)$. (In practice there is often massive ambiguity, and the number of parses may be too large to feasibly consider. Such cases require some form of pruning or approximation.)

Thus gradient ascent of the pseudolikelihood function is (at least approximately) computationally feasible. This is particularly useful since the Hessian of the logarithm of the pseudolikelihood function is non-positive, and therefore there are no local maxima. What's more, under mild conditions pseudolikelihood estimators (i.e. maximizers of (15)) are consistent [Chi, 1998].

4. Generalizations and other directions. There are a large number of extensions and applications of the grammatical tools just outlined. Treebank corpora, which consist of the hand-constructed parses of tens of thousands of sentences, are an extremely important resource for developing stochastic grammars [Marcus et al., 1993]. For example, the parses in a treebank can be used to generate, more or less automatically, a PCFG. Productions can be simply "read off" of the parse trees, and production probabilities can be estimated from relative frequencies, as explained in §3.1.2. Such PCFGs typically have on the order of 50 nonterminals and 15,000 productions. While the average number of parses per sentence is astronomical (we estimate greater than 10^{60}), the dynamic programming methods described in §3.2.3 are quite tractable, involving perhaps only hundreds of thousands of operations.

PCFGs derived from treebanks are moderately effective in parsing natural language [Charniak, 1996]. But the actual probabilities generated by these models (e.g. the probability of a given sentence) are considerably worse than those generated by other much simpler kinds of models, such as trigram models. This is presumably because these PCFGs ignore lexical dependencies between pairs or triples of words. For example, a typical treebank PCFG might contain the productions $\mathrm{VP} \to \mathrm{V}\,\mathrm{NP}, \mathrm{V} \to$ eat and $\mathrm{NP} \to$ pizza, in order to generate the string *eat pizza*. But since noun phrases such as *airplanes* are presumably also generated by productions such as $\mathrm{NP} \to$ airplanes, this grammar also generates unlikely strings such as *eat airplanes*.

One way of avoiding this difficulty is to *lexicalize* the grammar, i.e., to "split" the nonterminals so that they encode the "head" word of the phrase that they rewrite to. In the previous example, the corresponding lexicalized productions are $\mathrm{VP}_{\mathrm{eat}} \to \mathrm{V}_{\mathrm{eat}}\,\mathrm{NP}_{\mathrm{pizza}}, \mathrm{V}_{\mathrm{eat}} \to$ eat and $\mathrm{NP}_{\mathrm{pizza}} \to$ pizza. This permits the grammar to capture some of the lexical selectional preferences of verbs and other heads of phrases for specific head words. This technique of splitting the nonterminals is very general, and can be used to encode other kinds of nonlocal dependencies as well [Gazdar et al., 1985]. In fact, the state of the art probabilistic parsers can be regarded as PCFG parsers operating with very large, highly structured, nonterminals. Of course, this nonterminal splitting dramatically increases the number of nonterminals N and the number of productions R in the grammar, and this complicates both the computational problem [Eisner and Satta, 1999] and, more seriously, inference. While it is straight-forward to lexicalize

the productions of a context-free grammar, many or even most productions in the resulting grammar will not actually appear even in a large treebank. Developing methods for accurately estimating the probability of such productions by somehow exploiting the structure of the lexically split nonterminals is a central theme of much of the research in statistical parsing [Collins, 1996, Charniak, 1997].

While most current statistical parsers are elaborations of the PCFG approach just specified, there are a number of alternative approaches that are attracting interest. Because some natural languages are not context-free languages (as mentioned earlier), most linguistic theories of syntax incorporate context-sensitivity in some form or other. That is, according to these theories the set of trees corresponding to the sentences of a human language is not necessarily generated by a context-free grammar, and therefore the PCFG methods described above cannot be used to define a probability distribution over such sets of trees. One alternative is to employ the more general Gibbs models, discussed above in §3.3 (see for example [Abney, 1997]). Currently, approaches that apply Gibbs models build on previously existing "unification grammars" [Johnson et al., 1999], but this may not be optimal, as these grammars were initially designed to be used non-stochastically.

REFERENCES

[Abney et al., 1999] Abney, Steven, David McAllester, and Fernando Pereira. 1999. Relating probabilistic grammars and automata. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 542–549, San Francisco. Morgan Kaufmann.

[Abney, 1997] Abney, Steven P. 1997. Stochastic Attribute-Value Grammars. *Computational Linguistics*, **23**(4):597–617.

[Baum, 1972] Baum, L.E. 1972. An inequality and associated maximization techniques in statistical estimation of probabilistic functions of Markov processes. *Inequalities*, **3**:1–8.

[Besag, 1974] Besag, J. 1974. Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal of the Royal Statistical Society, Series D*, **36**:192–236.

[Besag, 1975] Besag, J. 1975. Statistical analysis of non-lattice data. *The Statistician*, **24**:179–195.

[Charniak, 1996] Charniak, Eugene. 1996. Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036, Menlo Park. AAAI Press/MIT Press.

[Charniak, 1997] Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Menlo Park. AAAI Press/MIT Press.

[Chi, 1998] Chi, Zhiyi. 1998. *Probability Models for Complex Systems*. PhD thesis, Brown University.

[Chi, 1999] Chi, Zhiyi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, **25**(1):131–160.

[Chi and Geman, 1998] Chi, Zhiyi and Stuart Geman. 1998. Estimation of probabilistic context-free grammars. *Computational Linguistics*, **24**(2):299–305.

[Chomsky, 1957] CHOMSKY, NOAM. 1957. *Syntactic Structures*. Mouton, The Hague.

[Collins, 1996] COLLINS, M.J. 1996. A new statistical parser based on bigram lexical dependencies. In *The Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, San Francisco. The Association for Computational Linguistics, Morgan Kaufmann.

[Culy, 1985] CULY, CHRISTOPHER. 1985. The complexity of the vocabulary of Bambara. *Linguistics and Philosophy*, **8**(3):345–352.

[Dempster et al., 1977] DEMPSTER, A., N. LAIRD, AND D. RUBIN. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, **39**:1–38.

[Eisner and Satta, 1999] EISNER, JASON AND GIORGIO SATTA. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 457–464, 1999.

[Foo, 1974] FOO, K.S. 1974. *Syntactic Methods in Pattern Recognition*. Academic Press.

[Foo, 1982] FOO, K.S. 1982. *Syntactic Pattern Recognition and Applications*. Prentice-Hall.

[Garey and Johnson, 1979] GAREY, MICHAEL R. AND DAVID S. JOHNSON. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

[Gazdar et al., 1985] GAZDAR, GERALD, EWAN KLEIN, GEOFFREY PULLUM, AND IVAN SAG. 1985. *Generalized Phrase Structure Grammar*. Basil Blackwell, Oxford.

[Grenander, 1976] GRENANDER, ULF. 1976. *Lectures in Pattern Theory. Volume 1: Pattern Synthesis*. Springer, Berlin.

[Harris, 1963] HARRIS, T.E. 1963. *The Theory of Branching Processes*. Springer, Berlin.

[Hopcroft and Ullman, 1979] HOPCROFT, JOHN E. AND JEFFREY D. ULLMAN. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

[Jelinek, 1997] JELINEK, FREDERICK. 1997. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts.

[Johnson et al., 1999] JOHNSON, MARK, STUART GEMAN, STEPHEN CANON, ZHIYI CHI, AND STEFAN RIEZLER. 1999. Estimators for stochastic "unification-based" grammars. In *The Proceedings of the 37th Annual Conference of the Association for Computational Linguistics*, pages 535–541, San Francisco. Morgan Kaufmann.

[Kaplan and Bresnan, 1982] KAPLAN, RONALD M. AND JOAN BRESNAN. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, Chapter 4, pages 173–281. The MIT Press.

[Kay et al., 1994] KAY, MARTIN, JEAN MARK GAVRON, AND PETER NORVIG. 1994. *Verbmobil: a translation system for face-to-face dialog*. CSLI Press, Stanford, California.

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

[Lari and Young, 1991] LARI, K. AND S.J. YOUNG. 1991. Applications of Stochastic Context-Free Grammars using the Inside-Outside algorithm. *Computer Speech and Language*, **5**:237–257.

[Marcus et al., 1993] MARCUS, MICHELL P., BEATRICE SANTORINI, AND MARY ANN MARCINKIEWICZ. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, **19**(2):313–330.

[Pollard and Sag, 1987] POLLARD, CARL AND IVAN A. SAG. 1987. *Information-based Syntax and Semantics*. Number 13 in CSLI Lecture Notes Series. Chicago University Press, Chicago.

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

[Shieber, 1986] SHIEBER, STUART M. 1986. *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes Series. Chicago University Press, Chicago.