Appears in: Carl Weir (ed.), Statistically-Based Natural Language Processing Techniques: Papers from the 1992 Workshop, pp. 20-27. Technical Report W-92-01, AAAI Press, Menlo Park, 1992.

A Probabilistic Parser and Its Application

Mark A. Jones
AT&T Bell Laboratories
600 Mountain Avenue, Rm. 2B-435
Murray Hill, NJ 07974–0636
jones@research.att.com

Jason M. Eisner
Emmanuel College, Cambridge
Cambridge CB2 3AP England
jme14@phoenix.cambridge.ac.uk

Abstract

We describe a general approach to the probabilistic parsing of context-free grammars. The method integrates context-sensitive statistical knowledge of various types (e.g., syntactic and semantic) and can be trained incrementally from a bracketed corpus. We introduce a variant of the GHR context-free recognition algorithm, and explain how to adapt it for efficient probabilistic parsing. In split-corpus testing on a real-world corpus of sentences from software testing documents, with 20 possible parses for a sentence of average length, the system finds and identifies the correct parse in 96% of the sentences for which it finds any parse, while producing only 1.03 parses per sentence for those sentences. Significantly, this success rate would be only 79% without the semantic statistics.

Introduction

In constrained domains, natural language processing can often provide leverage. At AT&T, for instance, NL technology can potentially help automate many aspects of software development. A typical example occurs in the software testing area. Here 250,000 English sentences specify the operational tests for a telephone switching system. The challenge is to to extract at least the surface content of this highly referential, naturally occurring text, as a first step in automating the largely manual testing process. The sentences vary in length and complexity, ranging from short sentences such as "Station B3 goes onhook" to 50 word sentences containing parentheticals, subordinate clauses, and conjunction. Fortunately the discourse is reasonably well focused: a large but finite number of telephonic concepts enter into a limited set of logical relationships. Such focus is characteristic of many sublanguages with practical importance (e.g., medical records).

We desire to press forward to NL techniques that are robust, that do not need complete grammars in advance, and that can be trained from existing corpora of sample sentences. Our approach to this problem grew out of earlier work [Jones et al 1991] on correcting the output of optical character recognition (OCR) systems. We were amazed at how much correction was possible using only low-level statistical knowledge about English (e.g., the frequency of digrams like "pa") and about common OCR mistakes (e.g., reporting "c" for "e"). As many as 90% of incorrect words could be fixed within the telephony sublanguage domain, and 70–80% for broader samples of English. Naturally we wondered whether more sophisticated uses of statistical knowledge could aid in such tasks as the one described above. The recent literature also reflects an increasing interest in statistical training methods for many NL tasks, including parsing [Jelinek and Lafferty 1991, Magerman and Marcus 1991, Bobrow 1991, Magerman and Weir 1992, Black, Jelinek, et al 1992], part of speech tagging [Church 1988], and corpora alignment [Dagan et al 1991, Gale and Church 1991].

Simply stated, we seek to build a parser that can construct accurate syntactic and semantic analyses for the sentences of a given language. The parser should know little or nothing about the target language, save what it can discover statistically from a representative corpus of analyzed sentences. When only unanalyzed sentences are available, a practical approach is to parse a small set of sentences by hand, to get started, and then to use the parser itself as a tool to suggest analyses (or partial analyses) for further sentences. A similar "bootstrapping" approach is found in [Simmons 1990]. The precise grammatical theory we use to hand-analyze sentences should not be crucial, so long as it is applied consistently and is not unduly large.

Parsing Algorithms

Following [Graham et al 1980], we adopt the following notation. An arbitrary context-free grammar is given by $G = (V, \Sigma, P, S)$, where V is the vocabulary of all symbols, Σ is the set of terminal symbols, P is the set of rewrite rules, and S is the start symbol. For an input sentence $w = a_1 a_2 \ldots a_n$, let $w_{i,j}$ denote the substring $a_{i+1} \ldots a_j$ and $w_i = w_{0,i}$ denote the prefix of length i. We use Greek letters $(\alpha, \beta, \ldots)$ to denote symbol strings in V^*.

Tabular dynamic programming algorithms are the methods of choice for ordinary context-free recognition [Cocke and Schwartz 1970, Earley 1970, Graham et al 1980]. Each entry $t_{i,j}$ in a table or chart, $\mathbf{t}$, holds a set of symbols or rules that *match* $w_{i,j}$. A symbol A matches $w_{i,j}$ if $A \Rightarrow^* w_{i,j}$. Some of these methods use *dotted rules* to represent progress in matching the input. For all $A \rightarrow \alpha\beta$ in P, $A \rightarrow \alpha \cdot \beta$ is a dotted rule of G. The dotted rule $A \rightarrow \alpha \cdot \beta$ matches $w_{i,j}$ (and hence is in the set $t_{i,j}$) if $\alpha \Rightarrow^* w_{i,j}$.

The dynamic programming algorithms work by combining shorter derivations into longer ones. In the CKY algorithm, the grammar is in Chomsky Normal Form. The symbol A may be added to $t_{j-1,j}$ by the lexical rule $A \rightarrow a_j$; or to $t_{i,j}$ by the rule $A \rightarrow BC$, if there exist symbols B in $t_{i,k}$ and C in $t_{k,j}$. In other words, CKY obeys the following invariant:

Invariant 1 *CKY: Add A to $t_{i,j}$ if and only if $A \Rightarrow^* w_{i,j}$.*

The principal drawback of the CKY method is that the algorithm finds matches that cannot lead to derivations for S. The GHR algorithm [Graham et al 1980] improves the average case performance by considering only matches that are consistent with the left context:

Invariant 2 *GHR: Add $A \rightarrow \alpha \cdot \beta$ to $t_{i,j}$ if and only if $\alpha \Rightarrow^* w_{i,j}$ and $S \Rightarrow^* w_i A \sigma$ for some $\sigma \in \Sigma^*$.*

In one sense, GHR seems to do as well as one could expect in an *on-line* recognizer that recognizes each prefix of w without lookahead. Still, the algorithm runs in time $O(n^3)$ and space $O(n^2)$ for arbitrary context-free grammars. Furthermore, in many applications the goal is not simply to recognize a grammatical sentence, but to find its possible parses, or the best parse. Extracting *all* parses from the chart can be quite expensive. Natural language constructs such as prepositional phrase attachments and noun-noun compounds can give rise to a Catalan number of parses [Winograd 1983], as in the classic sentence *"I saw a man in the park with a telescope."* With such inherent ambiguity, even refinements based on lookahead do not reduce the overall complexity. The only way to further improve performance is to find fewer parses—to track only those analyses that make semantic and pragmatic sense. Such an approach is not only potentially faster; it is usually more useful as well.

It is straightforward to turn the GHR recognizer into a chart parser. The chart will now store trees rather than dotted rules. Let $A \rightarrow \alpha_{i,j} \cdot \beta$ represent a *dotted tree* with root A that dominates $w_{i,j}$ $(i < j)$ through a contiguous sequence of child subtrees, $\alpha_{i,j}$. When the context is clear, we will refer to such a tree as $A_{i,j}$, or more generally as $h_{i,j}$. Here $\beta \in V^*$ as before; when β is null the tree is called *complete*.

We could modify Invariants 1 and 2 to refer to dotted trees. As in CKY, we could add a tree $A_{i,j}$ if and only if it dominated $w_{i,j}$. A stronger condition, similar to GHR, would further require $A_{i,j}$ to be syntactically and semantically consistent with the left context w_i. The problem remains, however, that the notion of contextual consistency is too weak—we want analyses that are contextually *probable*. Even semantic consistency is not enough. Many of the readings in the example above are internally consistent but still improbable. For example, it is possible that the example describes the sawing of a man, but not likely. To effectively reduce the search space, we must restrict our attention to analyses that are *probable* given the joint considerations of syntax, semantics, etc. We desire to form

Invariant 3 *Optimal (OPT): Add the dotted tree $A_{i,j}$ if and only if it is dominated by the "best parse tree" $\widehat{S}_{0,n}$, defined as the most probable complete tree of the form $S_{0,n}$.*

Of course, when we are parsing a new (unbracketed) sentence, $\widehat{S}_{0,n}$ is not known ahead of time. In a strictly left-right parser, without lookahead in the input string, it is generally impossible to guarantee that we only keep trees that appear as subtrees of $\widehat{S}_{0,n}$. Nevertheless, since language is generally processed by humans from left to right in real time, it is reasonable to suspect that the left-context contains enough information to severely limit nondeterminism. A first attempt might be

Invariant 4 *Most Probable (MP):*
Add $A_{i,j}$ if and only if $\sum_{\sigma \in \Sigma^*} \Pr[S \Rightarrow^* w_i A_{i,j} \sigma] \geq \sum_{\sigma \in \Sigma^*} \Pr[S \Rightarrow^* w_{i'} B_{i',j} \sigma]$ for all dotted trees $B_{i',j}$ that compete with $A_{i,j}$.

$A_{i,j}$ and $B_{i',j}$ are said to *compete* if they offer about the same level of explanation (see below) and neither dominates the other. Such trees are incompatible as explanations for a_j in particular, so only one can appear as part of $\widehat{S}_{0,n}$.

The MP criterion guesses the ultimate usefulness and probability of a dotted tree by considering only its left context. The left context may of course be unhelpful: for instance, there is no context at the beginning of the sentence. Worse, the left context may be misleading. In principle there is nothing wrong with this: even humans have difficulty with misleading "garden path" sentences. The price for being right and fast most of the time is the possibility of being fooled occasionally—as with any heuristic.

Even so, MP is too strict a criterion for most domains: it throws away many plausible trees, some of which may be necessary to build the preferred parse $\widehat{S}_{0,n}$ of the whole sentence. We modify MP so that instead of adding only the most likely tree in each set of competitors, it adds all trees within some fraction ϵ of the most likely one. Thus the parameter $0 \leq \epsilon < 1$ operationally determines the set of garden path sentences for the parser. If the left context is sufficiently misleading for a given ϵ, then useful trees may still be discarded. But in exchange for the certainty of producing every consistent analysis, we hope to find a good (statistically speaking) parse much faster by pruning away unlikely alternatives. If the number of alternatives is bounded by some constant k in practice, we can obtain an algorithm that is $O(n + ke)$ where e is the number of edges in the parse tree. For binary branching trees, $e = 2(n-1)$, and the algorithm is $O(n)$ as desired.

Invariant 5 *Reasonably Probable (RP): Add* $A_{i,j}$ *if and only if* $\sum_{\sigma \in \Sigma^*} \Pr[S \Rightarrow^* w_i A_{i,j} \sigma] \geq \epsilon \cdot \sum_{\sigma \in \Sigma^*} \Pr[S \Rightarrow^* w_{i'} B_{i',j} \sigma]$ *for all dotted trees* $B_{i',j}$ *that compete with* $A_{i,j}$*.*

An alternative approach would keep m competing rules from each set, where m is fixed. This has the advantage of guaranteeing a constant number of candidates, but the disadvantage of not adapting to the ambiguity level at each point in the sentence. The fractional method better reflects the kind of "memory load" effects seen in psycholinguistic studies of human parsing.

Algorithm 1 in Appendix A describes a parser that obeys the RP invariant. The algorithm returns the set of complete trees of the form $S_{0,n}$. We restrict the start symbol S to be non-recursive. If necessary a distinguished start symbol (e.g., $ROOT$) can be added to the grammar. Trees are created in three ways. First, the trivial trees a_j assert the presence of the input symbols. Second, the parser creates some "empty trees" from the grammar, although these are not added to the chart: they have the form $A \rightarrow \Lambda_{i,i} \cdot \beta$, where Λ denotes a sequence of zero subtrees. Third, the parser can combine trees into larger trees using the $\otimes$ operator. $\otimes$ pastes two adjacent trees together:

$$(A \rightarrow \alpha_{i,j} \cdot B\gamma) \otimes B_{j,k} = (A \rightarrow \alpha_{i,j} B_{j,k} \cdot \gamma)$$

Here the first argument of $\otimes$ must be an incomplete tree, while the second must be a complete tree. The operator can easily be extended to work with sets and charts:

$$Q \otimes R = \{A_{i,j} \otimes B_{j,k} \mid A_{i,j} \in Q \text{ and is incomplete, } B_{j,k} \in R \text{ and is complete}\}$$

$$\mathbf{t} \otimes R = (\bigcup t_{i,j}) \otimes R$$

Theorem 1 *No tree* $A_{i,j}$ *can dominate an incomplete tree* $B_{i',j}$*.*
Proof *Suppose otherwise. Then* $A_{i,j}$ *dominates the completion of* $B_{i',j}$*, and in particular some rightward extension* $B_{i',j} \otimes C_{j,k}$*, where* $C_{j,k}$ *is a complete tree with* $j < k$*. It follows that* $A_{i,j}$ *dominates* $w_{j,k}$*, a contradiction.* ■
Corollary *Given any two incomplete trees* $A_{i,j}$ *and* $B_{i',j}$*, neither dominates the other.* ■

Lemma 2 *In line 5 of Algorithm 1, no tree* $A_{i,j} \in N$ *can dominate a tree* $B_{i',j} \in N - E_j$*.*
Proof *Every tree in* $N - E_j$ *has just been created on this iteration of the* **while** *loop. But all subtrees dominated by* $A_{i,j}$ *were created on previous iterations.* ■

Theorem 3 *In line 5 of Algorithm 1, no tree* $A_{i,j} \in N$ *can dominate a tree* $B_{i',j} \in N$*.*
Proof *Either* $B_{i',j} \in N - E_j$*, or* $B_{i',j} \in E_j$ *and is incomplete. The claim now follows from the results above.* ■

The most important aspect of the tree-building process is the explicit construction and pruning of the set of competing hypotheses, N, during each iteration. It is here that the parser chooses which hypotheses to pursue. Theorem 3 states that the trees in N are in fact mutually exclusive. The algorithm is also careful to ensure that the trees in N are of roughly equal depth. Were this not the case, two equally likely deep trees might have to compete (on different iterations) with each other's subtrees. Since the shallow subtrees are always more probable than their ancestors ("the part is more likely than the whole"), this would lead to the pruning of both deep trees.

We now state a theorem regarding the parses that will be found without pruning.

Theorem 4 *In the special case of* $\epsilon = 0$*, Algorithm 1 computes precisely the derivations that could be extracted from the recognition sets of GHR up to position* j *in the input.*
Proof *When* $\epsilon = 0$*, only zero-probability dotted trees will be pruned. We assume that any parse tree permitted by the formal grammar G has probability* > 0*, as do its subtrees. Conversely,* $\Pr[A_{i,j} \rightarrow \alpha_{i,j} \cdot \beta \mid w_j] > 0$ *means that* $S \Rightarrow^* w_i \alpha_{i,j} \beta_{j,k} \gamma$ *is a valid derivation for some sequence of trees* $\beta_{j,k}$ *and some string* γ*.*

Thus $\Pr[A_{i,j} \rightarrow \alpha_{i,j} \cdot \beta \mid w_j] > 0$ *is equivalent to the statement that* $S \Rightarrow^* w_i A \sigma$ *for some* $\sigma = w_{j,k}\gamma \in \Sigma^*$*. Hence Invariant 5 adds* $A_{i,j} \rightarrow \alpha_{i,j} \cdot \beta$ *to the chart if and only if Invariant 2 adds* $A \rightarrow \alpha \cdot \beta$*.* ■

A significant advantage of the data-driven parser described by Algorithm 1 is its potential use in noisy recognition environments such as speech or OCR. In such applications, where many input hypotheses may compete, pruning is even more valuable in avoiding a combinatorial explosion.

Considerations in Probabilistic Parsing

Algorithm 1 does not specify a computation for $\Pr[h_{i,j} \mid w_j]$, and so leaves open several important questions:

1. What sources of knowledge (syntax, semantics, etc.) can help determine the probability of the dotted tree $h_{i,j}$?
2. What features of the left context w_i are relevant to this probability?

3. Given answers to the above questions, how can we compute $\Pr[h_{i,j} \mid w_j]$?
4. How much training is necessary to obtain sufficiently accurate statistics?

The answers are specific to the class of languages under consideration. For natural languages, reasonable performance can require a great deal of knowledge. To correctly interpret *"I saw a man in the park with a telescope,"* we may need to to know how often telescopes are used for seeing; how often a verb takes two prepositional phrases; who is most likely to have a telescope (me, the man, or the park); and so on.

Our system uses knowledge about the empirical frequencies of syntactic and semantic forms. However, our approach is quite general and would apply without modification to other knowledge sources, whether empirical or not.

The left-context probability $\Pr[h_{i,j} \mid w_j]$ depends on the literal input seen so far, w_j. How are we to know this probability if w_j is a novel string? As it turns out, we can compute it in terms of the left-context probabilities of other trees already in the chart, using *arbitrarily* weak independence assumptions. We will need empirical values for expressions of the form

$\Pr[h_{i,j} \otimes h_{j,k} \mid c_i \,\&\, h_{i,j} \,\&\, h_{j,k}]$

$\Pr[a_{i+1} \mid c_i]$

where c_i is one possible "partial interpretation" of w_i (constructed from other trees in the chart). If the language permits relatively strong independence assumptions, c_i need not be too detailed an interpretation; then we will not need too many statistics, or a large set of examples. On the other hand, if we refuse to make any independence assumptions at all, we will have to treat every string w_j as a special case, and keep separate statistics for every sentence of the language.

In the next section, we will outline the computation for a simple case where c_i contains no semantic information. In the final section we present a range of results, including cases in which syntactic and semantic information is jointly considered. For a further discussion of the syntactic and semantic representations and their probabilities, including a helpful example, see [Jones and Eisner 1992].

Note that semantic interdependence can operate across some distance in a sentence; in practice, the likelihood of $h_{i,j}$ may depend on even the earliest words of w_j. Compare *"The champagne was very bubbly"* with *"The hostess was very bubbly."* If we are to eliminate the incongruous meaning of "bubbly" in each case, we will need c_4 (a possible interpretation of the left context w_4) to indicate whether the subject of the sentence is human.

It remains an interesting empirical question whether it is more efficient (1) to compute highly accurate probabilities, via adequately detailed representations of left context, or (2) to use broader (e.g., non-semantic) representations, and compensate for inaccuracy by allowing more local nondeterminism. It can be cheaper to evaluate individual hypotheses under (2), and psycholinguistic evidence on on parallel lexical access [Tanenhaus et al 1985] may favor (2) for sublexical speech processing. On the other hand, if we permit too much nondeterminism, hypotheses proliferate and the complexity rises dramatically. Moreover, inaccurate probabilities make it difficult to choose among parses of an ambiguous sentence.

A Syntactic Probability Computation

Our parser constructs generalized syntactic and semantic representations, and so permits c_i to be as broad or as detailed as desired. Space prevents us from giving the general probability computation. Instead we sketch a simple but still useful special case that disregards semantics. Here the (mutually exclusive) descriptions c_i, where $0 \leq i < n$, will take the form "w_i is the kind of string that is followed by an *NP*" (or *VP*, etc.). We make our standard assumption that the probability of $h_{i,j}$ may depend on c_i, but is independent of everything else about w_i.[1] In this case, c_i is a function of the single correct incomplete dotted tree ending at i: so we are assuming that nothing else about w_i is relevant.

We wish to find $\Pr[h_{j,l} \mid w_l]$. Given $0 \leq j < n$, let E_j denote the subset of *incomplete* dotted trees in $\bigcup_i t_{i,j}$. We may assume that some member of E_j does appear in $\widehat{S}_{0,n}$. (When this assumption fails, $\widehat{S}_{0,n}$ will not be found no matter how accurate our probability computation is.) The corollary to Theorem 1 then implies that exactly one tree in E_j appears in $\widehat{S}_{0,n}$. We can therefore express $\Pr[h_{j,l} \mid w_l]$ as $\sum_{h_{i,j} \in E_j} \Pr[h_{i,j} \,\&\, h_{j,l} \mid w_l]$. We cache all the summands, as well as the sum, for future use.

So we only need an expression for $U = \Pr[h_{i,j} \,\&\, h_{j,l} \mid w_l]$. There are three cases. If $l = j+1$ and $h_{j,l} = a_{j+1}$, we can apply Bayes' Theorem:

$$\begin{aligned} U &= \Pr[h_{i,j} \,\&\, a_{j+1} \mid w_{j+1}] \\ &= \Pr[h_{i,j} \,\&\, a_{j+1} \mid w_j \,\&\, a_{j+1}] \\ &= \Pr[h_{i,j} \mid w_j \,\&\, a_{j+1}] \\ &= \frac{\Pr[a_{j+1} \mid h_{i,j} \,\&\, w_j] \cdot \Pr[h_{i,j} \mid w_j]}{\sum_{h_{i',j} \in E_j} \Pr[a_{j+1} \mid h_{i',j} \,\&\, w_j] \cdot \Pr[h_{i',j} \mid w_j]} \\ &= \frac{X_1 X_2}{\sum X_1' X_2'} \end{aligned}$$

In the second case, where $h_{j,l} = h_{j,k} \otimes h_{k,l}$, we factor as follows:

$$U = \Pr[h_{i,j} \,\&\, (h_{j,k} \otimes h_{k,l}) \mid w_l]$$

[1] In the language of classical statistics, we have a binary-valued random variable H that takes the value *true* iff the tree $h_{i,j}$ appears in the correct parse. We may treat the unknown p.d.f. for H as determined by the parameter w_i, the preceding input. Our assumption is that c_i is a sufficient statistic for w_i.

$$\begin{aligned} &= \Pr[h_{j,k} \otimes h_{k,l} \mid h_{i,j} \,\&\, h_{j,k} \,\&\, h_{k,l} \,\&\, w_l] \\ &\quad \cdot \Pr[h_{j,k} \,\&\, h_{k,l} \mid w_l] \cdot \Pr[h_{i,j} \mid h_{j,k} \,\&\, h_{k,l} \,\&\, w_l] \\ &= X_3 X_4 Y \end{aligned}$$

Insofar as our independence assumption holds, we can prove

$$Y \approx \Pr[h_{i,j} \mid h_{j,k} \,\&\, w_k] = \frac{\Pr[h_{i,j} \,\&\, h_{j,k} \mid w_k]}{\Pr[h_{j,k} \mid w_k]} = \frac{X_5}{X_6}$$

Finally, if $0 < j = l$ and $h_{j,j} = h_{j,l}$ is an empty tree, $A \rightarrow \Lambda_{j,j} \cdot \beta$ (X_5 sometimes yields this form):

$$\begin{aligned} U &= \Pr[h_{i,j} \,\&\, h_{j,j} \mid w_j] \\ &= \Pr[h_{i,j} \mid w_j] \cdot \Pr[h_{j,j} \mid h_{i,j} \,\&\, w_j] = X_7 X_8 \end{aligned}$$

Now X_2, X_2', X_4, X_5, X_6 and X_7 are all left-context probabilities for trees (and pairs of trees) that are already in the chart. In fact, all these probabilities have *already been computed and cached.*

X_1, X_1', X_3 and X_8, as well as the top-down probabilities $\Pr[\hat{S}_{0,0}]$, may be estimated from empirical statistics. Where $h_{i,j}$ is the incomplete tree $A \rightarrow \alpha_{i,j} \cdot B\beta$, define $c_j(h_{i,j})$ to be the symbol B. Thus if $h_{i,j}$ is correct, w_j is in fact the kind of string that is followed by a constituent of type $c_j(h_{i,j}) = B$. According to our independence assumption, nothing else about w_j (or $h_{i,j}$) matters. We therefore write

$X_1 \approx \Pr[a_{j+1} \mid c_j(h_{i,j})]$

$X_8 \approx \Pr[h_{j,j} \mid c_j(h_{i,j})]$,

and by similar assumptions (and abuse of notation),

$X_3 \approx \Pr[\otimes \mid c_k(h_{j,k}) \,\&\, \mathrm{root}(h_{k,l})]$.

An illustration may be helpful here. Suppose $A = h_{j,k}$ is the dotted tree $VP \rightarrow V_{j,k} \cdot NP$. Thus A has the property that $c_k(A) = NP$. Suppose further that $B = h_{k,l}$ is some complete tree representing an NP. The above statistic for X_3 gives the likelihood that such an A will bind such a B as its next child (rather than binding some deeper NP that dominates B). Our parser computes this statistic during training, simply by looking at the sample parse trees provided to it. GHR makes use of similar facts about the language, but deduces them from the formal grammar.

The key feature of this derivation (and of the more general version) is that it permits effective caching. Thanks to Bayes' Theorem, left-context probabilities are always written in terms of other, previously computed left-context probabilities. So $\Pr[h_{j,l} \mid w_l]$ can always be found by multiplying or adding together a few known numbers—regardless of the size of the dotted tree $h_{j,l}$.

If the size of the sets E_j is bounded by a constant, then the summations are bounded and each new tree can be evaluated in constant time. Since at most one member of E_j is actually in $\hat{S}_{0,n}$, the set may be kept small by effective pruning. Thus accurate probabilities have a double payoff. They permit us to prune aggressively, a strategy which both keeps the chart small and makes it easy to estimate the probabilities of its entries.

Status and Results

Our parser serves as a component of the software testing application mentioned in the introduction (for details, see [Nonnenmann and Eddy 1992] and [Jones and Eisner 1992]). It has been trained on sample parse trees for over 400 sentences in the domain. The trees use lexical tags from the Brown Corpus [Francis and Kucera 1982] and fairly traditional phrase structure labels (S, NP, etc.). Although the "telephonese" sublanguage is an unrestricted subset of English, it differs statistically from English taken as a whole. The strength of trainable systems is their ability to adapt to such naturally occurring (and evolving) sublanguages.

The training corpus contains 308 distinct lexical items which participate in 355 part of speech rules. There are 55 distinct nonterminal labels, including 35 parts of speech. Sentences range from 5 to 47 words in length (counting punctuation). The average sentence is 11 long; the average parse tree is 9 deep with 31 nonterminal nodes.

We take our grammar to be the smallest set of symbols and context-free rules needed to write down every tree in the corpus. The *corpus perplexity* $b(C)$ measures the ambiguity of any set of sentences C under a given grammar:

$$\log b(C) = \frac{\sum_{S \in C} \log(\text{number of parses for } S)}{\sum_{S \in C} \text{number of words in } S}$$

Using GHR to parse the corpus exhaustively, we measure $b(C) = 1.313$. Thus a typical 11-word sentence has $1.313^{11} \approx 20$ parses, only one of which is correct.[2] 10% of the sentences have more than 1400 possible parses each.

Our working parser uses a hand-coded translation function τ to construct the semantic interpretations of dotted trees. It also makes use of some formal improvements to the algorithm, which have been omitted here for space reasons.

To date, we have tried to address three questions. First, particularly when the parser has enough knowledge to generate at least one parse, can it generate and identify the *correct* parse? Second, how quickly is the parser accumulating the knowledge necessary to get at least one parse? Third, how effectively does the algorithm control the combinatorial proliferation of hypotheses?

Accuracy

We have done several experiments to measure the accuracy of the parser on untrained sentences. Figure 1 summarizes two such experiments, which tested the accuracy of (i) joint syntactic-semantic statistics, and (ii) syntactic statistics alone (as formulated above). To best use our rather small set of 429 bracketed

[2] [Black, Jelinek, et al 1992], who call $b(C)$ the "parse base," report almost identical numbers for an corpus of sentences from computer manuals.

	(i) joint syntax and semantics	(ii) syntax alone
%(some parse found)	81%	76%
%(top parse correct | some parse found)	96%	79%
#(parses/sentence | some parse found)	1.03	1.33
%(some parse first found at 10^{-1})	53%	30%
%(some parse first found at 10^{-2})	19%	29%
%(some parse first found at 10^{-3})	9%	17%

Figure 1: Benefits of semantic knowledge.

sentences, we trained on each 428 sentence subset and tested on the remaining sentence. Each sentence was parsed with progressively wider search beams of $\epsilon = 10^{-1}, 10^{-2}$, and 10^{-3}, until at least one parse was found.

We scored a parse as correct only if it matched the target parse tree exactly. For example, we would disallow a parse that was in error on a part-of-speech tag, a prepositional attachment, or the internal structure of a major constituent. Some other papers have used less stringent scoring methods (e.g., [Black, Jelinek, et al 1992]), sometimes because their corpora were not fully bracketed to start with.

In experiment (i), using joint syntactic and semantic statistics, the parser correctly generated the target parse as its top choice in 96% of the cases where it found any parse. It achieved this rate while generating only 1.03 parses for each of those sentences. For 53% of the test sentences, the parser found one or more parses at the narrowest beam level, 10^{-1}. For 19%, the first parse was found at 10^{-2}. For another 9%, the first parse was found at 10^{-3}. For the remaining 19% of the test sentences, no parse was found by 10^{-3}; half of these contained unique vocabulary not seen in the training data.

The contrast between experiments (i) and (ii) indicates the importance of the statistical independence assumptions made by a language model. In experiment (ii), the left context still generated syntactic expectations that influenced the probabilities, but the parser assumed that the semantic role-filler assignments established by the left context were irrelevant. In contrast to the 96% accuracy yielded by the joint syntactic and semantic statistics of experiment (i), these syntax-only statistics picked the target parse in only 79% of the sentences with some parse. The weaker statistics in (ii) also forced the parser to use wider beams.

Knowledge Convergence

After 429 sentences in the telephony domain, the grammar is still growing: the rate of new "structural" (syntactic) rules is diminishing rapidly, but the vocabulary continues to grow significantly. In an attempt to determine how well our incomplete statistics are generalizing, we ran three experiments, based on a 3-way split, a 10-way split and a 429-way split of the corpus. In the 10-way split, for example, we tested on each tenth after training on the other nine-tenths. The parser used joint syntactic and semantic statistics for all three experiments. The results are summarized in Figure 2.

When the parser was able to find at least one parse, its top choice was nearly always correct (96%) in each experiment. However, the chance of finding at least one parse within $\epsilon = 10^{-3}$ increased with the amount of training. The overall success rates were 70% (3-way split), 76% (10-way split), and 77% (429-way split). In each case, a substantial fraction of the error is attributable to test sentences containing words that had not appeared in the training. The remaining error represents grammatically novel structures and/or an undertrained statistical model.

We would also like to know what the parser's accuracy will be when we are very well trained. In a fourth experiment, we trained on all 429 sentences before parsing them. Here, the parser produced only 1.02 parses/sentence and recovered the target parse 98.4% of the time. The correct parse was always ranked first whenever it was found. On the 428 sentences that had at least one parse, the parser only failed to find 10 of the 12,769 target constituents. For the present small corpus, there is apparently no substitute for having seen the test sentence itself.

The fourth experiment, unlike the other three, demonstrates the validity of the independence assumptions in our statistical model. It shows that for this corpus, the parser's generalization performance does not come at the expense of "memorization." That is, the statistics retain enough information for the parser to accurately reconstruct the training data.

Performance

For those sentences where some parse was found at beam 10^{-1} during experiment (i) above, 74% of the arcs added to the chart were actually required. We call this measure the *focus* of the parser since it quantifies how much of the work done was necessary. The *constituent focus*, or percentage of the completed constituents that were necessary, was even higher—78%. The constituent focus fell gradually with wider beams, to 75% at 10^{-2} and 65% at 10^{-3}.

		3-way split	10-way split	429-way split	Test on training
%(top parse correct	some parse found)	96%	96%	96%	98.6%
%(top parse correct	no unknown words)	81%	84%	85%	98.4%
%(top parse correct)		70%	76%	77%	98.4%

Figure 2: Effect of increased training on accuracy.

To remove the effect of the pruning schedule, we tried rerunning experiment (i) at a constant beam width of $\epsilon = 10^{-3}$. Here the constituent focus was 68% when some parse was found. In other words, a correct constituent had, on average, only half a competitor within three orders of magnitude.

In that experiment, the total number of arcs generated during a parse (before pruning) had a 94% linear correlation with sentence length. The shorter half of the corpus ($\leq$ 9 words) yielded the same regression line as the longer half. Moreover, a QQ-plot showed that the residuals were well-modeled by a (truncated) Gaussian distribution. These results suggest $O(n)$ time and space performance on this corpus, plus a Gaussian noise factor.[3]

Extensibility

Hand-coded natural language systems tend to be plagued by the potential open-endedness of the knowledge required. The corresponding problem for statistical schemes is undertraining. In our task, we do not have a large set of analyzed examples, or a complete lexicon for telephonese. To help compensate, the parser can utilize hand-coded knowledge as well as statistical knowledge. The hand-coded knowledge expresses general knowledge about linguistic subtheories or classes of rules, not specific knowledge about particular rules.

As an example, consider the problem of assigning part of speech to novel words. Several sources of knowledge may help suggest the correct part of speech class: the state of the parser when it encounters the novel word, the relative closedness of the class, the morphology of the word, and orthographic conventions like capitalization. An experimental version of our parser combines various forms of evidence to assign probabilities to novel lexical rules. Using this technique, the experimental parser can take a novel sentence such as "XX YY goes ZZ" and derive syntactic and semantic representations analogous to "Station B1 goes offhook."

[3] In keeping with usual statistical practice, we discarded the two 47-word outliers; the remaining sentences were 5–24 words long. We also discarded the 88 sentences with unknown words and/or no parses found, leaving 339 sentences.

Conclusions and Future Work

Truly robust natural language systems will require both the distributional knowledge and the general linguistic knowledge that are available to humans. Such knowledge will help these systems perform quickly and accurately, even under conditions of noise, ambiguity, or novelty. We are especially interested in bootstrapping approaches to enable a parser to learn more directly from an unbracketed corpus. We would like to combine statistical techniques with weak prior theories of syntax, semantics, and/or low-level recognition such as speech and OCR. Such theories provide an environment in which language learning can take place.

References

[Black, Jelinek, et al 1992] Black, E., Jelinek, F., Lafferty, J., Magerman, D.M., Mercer, R., and Roukos, S. "Towards History-Based Grammars: Using Richer Models for Probabilistic Parsing," *Fifth DARPA Workshop on Speech and Natural Language Processing.* February 1992.

[Bobrow 1991] Bobrow, R.J. "Statistical Agenda Parsing," *Fourth DARPA Workshop on Speech and Natural Language Processing.* February 1991.

[Cocke and Schwartz 1970] Cocke, J. and Schwartz, J.I. *Programming Languages and Their Compilers.* Courant Institute of Mathematical Sciences, New York University, New York, 1970.

[Church 1988] Church, K. W. "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text," *Proc. of the Second Conference on Applied Natural Language Processing.* Austin, Texas, 1988, pp. 136-143.

[Dagan et al 1991] Dagan, I., Itai, A., Schwall, U. "Two Languages Are More Informative Than One," *Proc. of the 29th Annual Meeting of the Association for Computational Linguistics.* Berkeley, California, 1991, pp. 130-137.

[Earley 1970] Earley, J. "An Efficient Context-Free Parsing Algorithm," *Communications of the ACM 13 (2).* February 1970, pp. 94-102.

[Francis and Kucera 1982] Francis, W. and Kucera, H. *Frequency Analysis of English Usage.* Houghton Mifflin, Boston, 1982.

[Gale and Church 1991] Gale, W. A. and Church, K. W. "A Program for Aligning Sentences in Bilingual

Corpora," *Proc. of the 29th Annual Meeting of the Association for Computational Linguistics.* Berkeley, California, 1991, pp. 177-184.

[Graham et al 1980] Graham, S.L., Harrison, M.A. and Ruzzo, W.L. "An Improved Context-Free Recognizer," *ACM Transactions on Programming Languages and Systems 2 (3).* 1980, pp. 415-463.

[Jelinek and Lafferty 1991] Jelinek, F. and Lafferty, J.D. "Computation of the Probability of Initial Substring Generation by Stochastic Context-Free Grammars," *Computations Linguistics 17 (3).* 1991, pp. 315-323.

[Jones and Eisner 1992] Jones, M.A. and Eisner, J. "A Probabilistic Parser Applied to Software Testing Documents," *Tenth National Conference on Artificial Intelligence*, San Jose, CA, 1992.

[Jones et al 1991] Jones, M.A., Story, G.A., and Ballard, B.W. "Using Multiple Knowledge Sources in a Bayesian OCR Post-Processor," *First International Conference on Document Analysis and Retrieval.* St. Malo, France, September 1991, pp. 925-933.

[Magerman and Marcus 1991] Magerman, D.M. and Marcus, M.P. "$\mathcal{P}$earl: A Probabilistic Chart Parser," *Fourth DARPA Workshop on Speech and Natural Language Processing.* February 1991.

[Magerman and Weir 1992] Magerman, D.M. and Weir, C. "Probabilistic Prediction and $\mathcal{P}$icky Chart Parsing," *Fifth DARPA Workshop on Speech and Natural Language Processing.* February 1992.

[Nonnenmann and Eddy 1992] Nonnenmann, U., and Eddy J.K. "KITSS - A Functional Software Testing System Using a Hybrid Domain Model," *Proc. of 8th IEEE Conference on Artificial Intelligence Applications*, Monterey, CA, March 1992.

[Simmons 1990] Simmons, R. and Yu, Y. "The Acquisition and Application of Context Sensitive Grammar for English," *Proc. of the 29th Annual Meeting of the Association for Computational Linguistics.* Berkeley, California, 1991, pp. 122-129.

[Tanenhaus et al 1985] Tanenhaus, M.K., Carlson, G.N. and Seidenberg, M.S. "Do Listeners Compute Linguistic Representations?" in *Natural Language Parsing* (eds. D. R. Dowty, L. Karttunen and A.M. Zwicky). Cambridge University Press, 1985, pp. 359-408.

[Winograd 1983] Winograd, T. *Language as a Cognitive Process, Volume 1: Syntax.* Addison-Wesley, 1983.

Appendix A: The Parsing Algorithm

Algorithm 1. PARSE(w):

(* create an $(n+1) \times (n+1)$ chart $\mathbf{t} = (t_{i,j})$: *)
  $t_{0,0} := \{S \rightarrow \Lambda_{0,0} \cdot \alpha \mid S \rightarrow \alpha$ is in $P\}$;
  for j := 1 **to** n **do**
    $D_j := t_{j-1,j} := \{a_j\}$; $E_j := \emptyset$;
    while $D_j \neq \emptyset$
      $N := (\mathbf{t} \otimes D_j) \cup (\text{PREDICT}(D_j) \otimes D_j) \cup E_j$;
      $R :=$PRUNE(N);
      $D_j := E_j := \emptyset$;
      for $h_{i,j} \in R$ **do**
        $t_{i,j} := t_{i,j} \cup \{h_{i,j}\}$;
        if $h_{i,j}$ is complete
          then $D_j := D_j \cup \{h_{i,j}\}$
          else $E_j := E_j \cup \{h_{i,j}\}$
      endfor
    endwhile
  endfor;
  return {all complete S-trees in $t_{0,n}$}

Function PREDICT(D):

  return $\{C \rightarrow \Lambda_{i,i} \cdot A\beta \mid C \rightarrow A\beta$ is in P, some complete tree $A_{i,j}$ is in D, and $C \neq S\}$

Function PRUNE(N):

(* only likely trees are kept *)
  $R := \emptyset$;
  $threshold := \epsilon \cdot \max_{h_{i,j} \in N} \Pr(h_{i,j} | w_{0,j})$;
  for $h_{i,j}$ **in** N **do**
    if $\Pr(h_{i,j} | w_{0,j}) > threshold$
      then $R := R \cup \{h_{i,j}\}$;
  endfor;
  return R