Proceedings of National Conference on Artificial Intelligence (AAAI-92), San Jose, 1992, pp. 322-328.

A Probabilistic Parser Applied to Software Testing Documents

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 an approach to training a statistical parser from a bracketed corpus, and demonstrate its use in a software testing application that translates English specifications into an automated testing language. A grammar is not explicitly specified; the rules and contextual probabilities of occurrence are automatically generated from the corpus. The parser is extremely successful at producing and identifying the correct parse, and nearly deterministic in the number of parses that it produces. To compensate for undertraining, the parser also uses general, linguistic subtheories which aid in guessing some types of novel structures.

GOAL: Activate CFA [call forwarding] using CFA Activation Access Code.

ACTION: Set station B2 without redirect notification. Station B2 goes offhook and dials CFA Activation Access Code.

VERIFY: Station B2 receives the second dial tone.

ACTION: Station B2 dials the extension of station B3.

VERIFY: Station B2 receives confirmation tone. The status lamp associated with the CFA button at B2 is lit.

VERIFY: ...

Figure 1: An Example Test Case

Introduction

In constrained domains, natural language processing can often provide leverage. In software testing at AT&T, for example, 20,000 English test cases prescribe the behavior of a telephone switching system. A test case consists of about a dozen sentences describing the goal of the test, the actions to perform, and the conditions to verify. Figure 1 shows part of a simple test case. Current practice is to execute the tests by hand, or else hand-translate them into a low-level, executable language for automatic testing. Coding the tests in the executable language is tedious and error-prone, and the English versions must be maintained anyway for readability.

We have constructed a system called KITSS (Knowledge-Based Interactive Test Script System), which can be viewed as a system for machine-assisted translation from English to code. Both the English test cases and the executable target language are part of a pre-existing testing environment that KITSS must fit into. The basic structure of the system is given in Figure 2. English test cases undergo a series of translation steps, some of which are interactively guided by a tester.

The *completeness and interaction analyzer* is the pragmatic component that understands the basic axioms and conventions of telephony. Its task is to flesh out the test description provided by the English sentences. This is challenging because the sentences omit many implicit conditions and actions. In addition, some sentences ("Make B1 busy") require the analyzer to create simple plans. The analyzer produces a formal description of the test, which the back-end *translator* then renders as executable code. A more complete description of the goals of the system, its architecture and the software testing problem can be found in [Nonnenmann and Eddy 1992].

This paper discusses the *natural language processor* or linguistic component, which must extract at least the surface content of a highly referential, naturally occurring text. The sentences vary in length, ranging from short sentences such as "Station B3 goes onhook" to 50 word sentences containing parentheticals, subordinate clauses, and conjunction. The principal leverage is that the discourse is reasonably well focused: a large, but finite, number of telephonic concepts enter into a finite set of relationships.

Natural Language Processing in KITSS

The KITSS linguistic component uses three types of knowledge to translate English sentences quickly and accurately into a logical form:

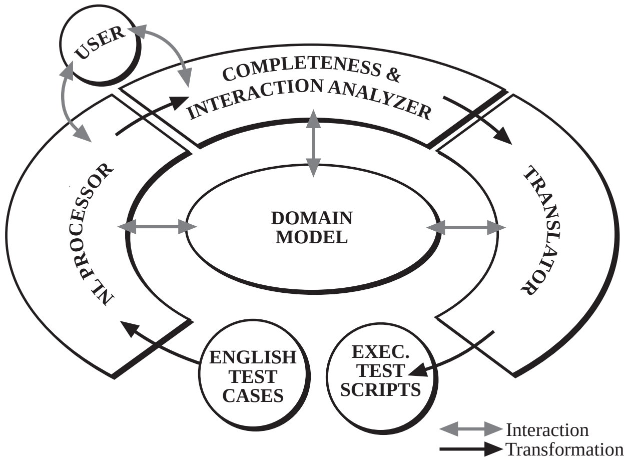

Figure 2: KITSS Architecture

1. *syntactic:* empirical statistics about common constructions
2. *semantic:* empirical statistics about common concepts
3. *referential:* expert knowledge about the logical representation of concepts

Figure 3 illustrates the syntactic, semantic, and logical representations computed for one analysis of the sentence "Place a call from station B1 to station B2." We will not say much here about the referential knowledge that finally rewrites the surface semantic representation as temporal logic. KITSS currently uses a hand-coded production system that includes linguistic rules (e.g., active-passive and conjunction), discourse rules (e.g., definite reference), and domain-specific canonicalization rules.

An undirected parser would generate many alternative (incorrect) hypotheses regarding the structure and interpretation of the sentence in Figure 3. It might try attaching the prepositional phrases to the noun phrase "a call," or treating "to station B2" as an infinitive phrase. In designing a parsing technique for the KITSS system, we wanted to exploit the statistical regularities that make one interpretation far likelier than others. In line with our earlier success on statistical error-correction for optical character recognizers (OCR devices) [Jones et al 1991], we sought ways to "bootstrap" the acquisition of statistical domain knowledge—in this case, knowledge about the likelihood of syntactic and semantic substructures in the test case sentences.

Note that initially we may have only a corpus of raw sentences, not a corpus of their target syntactic and semantic structures. While it is impractical to hand-analyze a large portion of the corpus, it is possible to do a relatively small number 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 approach to training is found in [Simmons 1991]. We will assume below that we have access to a training set of syntactic and semantic structures.[1]

Issues in Probabilistic Parsing

To generalize from the training corpus to new sentences, we will need to induce a good statistical model of the language. But the statistical distributions in a natural language reflect a great many factors, some of them at odds with each other in unexpected ways. Chomsky's famous sentence, "Colorless green ideas sleep furiously," is syntactically quite reasonable—but semantic nonsense—but, for historical reasons, quite common in conference papers. Or consider the classic illustration of attachment ambiguity: "I saw a man in the park with a telescope." One interpretation of this sentence holds that I used a telescope to see a man. To judge the relative likelihood, we may want to know how often telescopes are used for "seeing" (vs. "sawing"); 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.

Thus many features of a sentence may be significant. Within a restricted domain such as KITSS, the distributions are further shaped by the domain subject matter and by stylistic conventions. Sentences such as "Station B3 goes onhook" may be rare in the newspaper but common in the KITSS application. For stations to "go" is a test script idiom.

We want our statistics to capture more than the syntactic correlations. Our strategy is to build up rich interpretations of the sentences as we are parsing them. We take care to interpret every subtree that we generate. Thus, when we are deciding whether to combine two subtrees later on, we will know what the subtrees "mean." Furthermore, we will have semantic readings for other, possibly relevant portions of the sentence.

The semantic information helps to expose deep similarities and deep differences among sentences. Two trees that are semantically similar are likely to combine in similar ways. With semantic interpretations, we directly represent the fact that in one hypothesis the telescope is used for seeing. This fact is obscured in the corresponding syntactic tree—and even more so in the original sentence, where "saw" and "telescope" appear far apart.

Formally, let Ω be a given space of possible *interpretations*. We model a phrase structure rule, $L^k \rightarrow$

[1] In KITSS, only the syntactic bracketing is ever fully manual. The system automatically constructs a semantics for each training example from its syntax, using a set of translation rules. Most of these rules are inferred from a default theory of syntactic-semantic type correspondences.

```
String:    Place a call from station B1 to station B2 .
Syntax:    (SP (S (VP (VP (VP (VB "Place") (NP (AT "a") (NN "call")))
                          (PP (IN "from") (NP (NN "station") (NPR "B1"))))
                      (PP (IN "to") (NP (NN "station") (NPR "B2")))))
               (\. "."))
Semantics: (PLACE (:OBJECT (CALL (:NUMBER SING) (:REF A)))
                  (:FROM (STATION (:NUMBER SING) (:NAME "B1")))
                  (:TO (STATION (:NUMBER SING) (:NAME "B2")))
                  (:MOOD DECL) ...)
Logic:     ((OCCURS (PLACES-CALL B1 B2 CALL-812)))
```

Figure 3: Representations Formed in Processing a Sentence

$R_1\, R_2 \ldots R_m$, as an m-ary function taking values in Ω. $R_1 \ldots R_m$ give type restrictions on the m arguments. The function describes how to build an interpretation of type L from m contiguous substring interpretations of types $R_1 \ldots R_m$. The rule number k distinguishes rules that have the same domain and range, but differ functionally (e.g., in the case of a noun with two meanings); the chart maintains distinct hypotheses for the alternative interpretations.

In practice, our Ω consists of joint syntactic-semantic interpretations. The syntactic half of an interpretation is simply the parse tree. The semantic half is built compositionally from lexically-derived heads, slots and fillers in a standard frame language, as illustrated in Figure 3.

Experiments confirm the value of this approach for statistical parsing. When we run our parser with semantics turned off, its *syntactic* accuracy rate drops from 99% to 66%, and it runs far more slowly.

The KITSS Algorithm

The KITSS parsing algorithm (given as Algorithm 1 in Appendix A) is a variant of tabular or chart parsing methods for context-free languages [Cocke and Schwartz 1970, Earley 1970, Graham et al 1980]. It scans the sentence from left to right, assembling possible partial interpretations of the sentence; but it continually discards interpretations that are statistically unlikely.

The grammar rules and statistics are generated automatically by training on a bracketed corpus. The grammar is taken to be the smallest set of symbols and rules needed to write down all the parse trees in the corpus. The statistics are context-sensitive; they concern the frequencies with which the interpreted subtrees co-occur. Incremental training is permitted. The model is that the system considers a new sample sentence, updates its database, and throws the sentence away.

A 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 rules, and S is the start symbol. The start symbol is restricted to be non-recursive. A distinguished start symbol (e.g., $ROOT$) can be added to the grammar if necessary. For an input sentence $w = a_1 a_2 \ldots a_{|w|}$ $(a_i \in \Sigma)$, let $w_{i,j}$ denote the substring $a_{i+1} \ldots a_j$. For example, $w_{0,3}$ denotes the first three words.

The algorithm operates bottom-up from left to right in the input string. At each point j in the input string, the algorithm constructs hypotheses about the immediately preceding substrings $w_{i,j}$. A *complete hypothesis* is a parse tree for some substring of the sentence; we write it as $[L\, r_1\, r_2 \ldots r_m]$, where $L \rightarrow R_1\, R_2 \ldots R_m$ is a rule in P and each subtree r_i is itself a complete hypothesis with root R_i. An *incomplete hypothesis* is similar, except it is missing one or more of its rightmost branches. We write it as $[L\, r_1\, r_2 \ldots r_q\, +], 0 \leq q < m$.

We use the notation $h_{i,j}$ to refer to a hypothesis that dominates the string $w_{i,j}$. If a hypothesis $h_{i,j}$ is judged to be likely, it is added to a set $t_{i,j}$ in a $(|w| + 1) \times (|w| + 1)$ chart $\mathbf{t}$.

"Empty" hypotheses, which are created directly from the grammar, have the form $[L\, +]$. "Input" hypotheses just assert the existence of a_i and are complete; these are usually assigned probability 1, but normalized sets of input hypotheses could be used in noisy recognition environments such as speech or OCR.

Longer hypotheses are created by the $\otimes$ operator, which attaches a new child to a tree. The $\otimes$ product of two hypotheses is the smallest set respecting the condition that

$$\text{whenever} \left\{ \begin{array}{l} h_{i,j} = [L\, r_1 \ldots r_q\, +], \\ h_{j,k} = r_{q+1}, \text{ and} \\ (L \rightarrow R_1 \ldots R_q\, R_{q+1} \ldots R_m) \in P \end{array} \right.$$

$$\text{then} \left\{ \begin{array}{ll} [L\, r_1 \ldots r_q\, r_{q+1}\, +] \in (h_{i,j} \otimes h_{j,k}) & \text{if } q+1 < m \\ [L\, r_1 \ldots r_m] \in (h_{i,j} \otimes h_{j,k}) & \text{if } q+1 = m \end{array} \right.$$

Note that $\otimes$ returns a set of 0, 1, or 2 hypotheses. The first argument of $\otimes$ is ordinarily an incomplete hypothesis, while the second is a complete hypothesis immediately to its right. Otherwise $\otimes$ returns the empty set. The operator can easily be extended to act on sets and charts:

$$Q \otimes R := \bigcup \{h \otimes h' \mid h \in Q,\ h' \in R\}$$
$$\mathbf{t} \otimes R := (\bigcup t_{i,j}) \otimes R$$

The algorithm returns the set of complete hypotheses in $t_{0,|w|}$ whose roots are S, the start symbol. Each of these parses has an associated probability.

$$\begin{aligned} V &= \{\mathit{VP},\ \mathit{V},\ \text{“\textit{Place}”},\ \mathit{NP},\ \mathit{Det},\ \text{“\textit{a}”},\ \mathit{N},\ \text{“\textit{call}”}\ldots\} \\ \Sigma &= \{\text{“\textit{Place}”},\ \text{“\textit{a}”},\ \text{“\textit{call}”}\ldots\} \\ P &= \{\mathit{VP} \to \mathit{V}\ \mathit{NP},\ \mathit{NP} \to \mathit{Det}\ \mathit{N},\ \mathit{V} \to \text{“\textit{Place}”}, \\ &\quad\ \mathit{Det} \to \text{“\textit{a}”},\ \mathit{N} \to \text{“\textit{call}”}\ldots\} \\ S &= \mathit{VP} \end{aligned}$$

```
             VP
            /  \
           /    NP
          /    /  \
         V   Det   N
         |    |    |
   "Place" "a"  "call"
```

Figure 4: A Parse Tree for $w =$ *“Place” “a” “call”*

During the parsing process, a *left-context probability* or LCP, $\Pr(h_{i,j} \mid w_{0,j})$, is used to prune sets of competing hypotheses. Pruning severity depends on the *beam width*, $0 < \epsilon \leq 1$. A beam width of 10^{-2} keeps only those alternative hypotheses that are judged at least 1% as likely as the leading contender in the set. The correct parse can survive only if all of its constituent hypotheses meet this criterion; thus ϵ operationally determines the set of garden path sentences for the parser. If any correct hypothesis is pruned, then the correct parse will not be found (indeed, perhaps no parse will be found). This can happen in garden path sentences. It may also happen if the statistical database provides an inadequate or incomplete model of the language.

Probability Calculations

The parsing algorithm keeps or discards a hypothesis according to the left-context probability $\Pr(h_{i,j} \mid w_{0,j})$. The more accurate this value, the better we will do at pruning the search space. How can we compute it without assuming context-freeness?

We are able to decompose the probability into a product of corpus statistics (which we look up in a fixed hash table) and the LCPs of other hypotheses (which we computed earlier in the parse). Space prevents us from giving the formal derivation. Instead we will work through part of an example.

Figure 4 gives a small grammar fragment, with a possible parse tree for a short sentence. For convenience, we will name various trees and subtrees as follows:

$$\begin{aligned} \mathit{place} &= \text{“\textit{Place}”}_{0,1} \\ a &= \text{“\textit{a}”}_{1,2} \\ \mathit{call} &= \text{“\textit{call}”}_{2,3} \\ v &= [V\ \text{“\textit{Place}”}]_{0,1} \\ \mathit{det} &= [\mathit{Det}\ \text{“\textit{a}”}]_{1,2} \\ n &= [N\ \text{“\textit{call}”}]_{2,3} \\ \mathit{np}^1 &= [\mathit{NP}\ [\mathit{Det}\ \text{“\textit{a}”}]\ +]_{1,2} \\ \mathit{np} &= [\mathit{NP}\ [\mathit{Det}\ \text{“\textit{a}”}]\ [N\ \text{“\textit{call}”}]]_{1,3} \\ \mathit{vp}^1 &= [\mathit{VP}\ [V\ \text{“\textit{Place}”}]\ +]_{0,1} \\ \mathit{vp} &= [\mathit{VP}\ [V\ \text{“\textit{Place}”}] \\ &\qquad [\mathit{NP}\ [\mathit{Det}\ \text{“\textit{a}”}]\ [N\ \text{“\textit{call}”}]]]_{0,3} \end{aligned}$$

These trees correspond to the hypotheses of the previous section. Note carefully that—for example—the tree $\mathit{vp} \in \mathit{vp}^1 \otimes \mathit{np}$ is correct if and only if the trees vp^1 and np are also correct. We will use this fact soon.

Left-Context Probabilities

We begin with some remarks about $\Pr(\mathit{np} \mid w_{0,3})$, the LCP that np is the correct interpretation of $w_{1,3}$. This probability depends on the first word of the sentence, $w_{0,1}$, and in particular on the interpretation of $w_{0,1}$. (For example: if the statistics suggest that “Place” is a noun rather than a verb, the np hypothesis may be unlikely.) The correct computation is

$$\begin{aligned} \Pr(\mathit{np} \mid w_{0,3}) &= \Pr(\mathit{vp}^1 \,\&\, \mathit{np} \mid w_{0,3}) \\ &\quad + \Pr(X \,\&\, \mathit{np} \mid w_{0,3}) \\ &\quad + \Pr(Y \,\&\, \mathit{np} \mid w_{0,3}) + \ldots \end{aligned} \tag{1}$$

where $\mathit{vp}^1, X, Y, \ldots$ are a set of (mutually exclusive) possible explanations for “Place.” The summands in equation 1 are typical terms in our derivation. They are LCPs for *chains* of one or more contiguous hypotheses.

Now let us skip ahead to the end of the sentence, when the parser has finished building the complete tree vp. We decompose this tree's LCP as follows:

$$\begin{aligned} \Pr(\mathit{vp} \mid w_{0,3}) &= \Pr(\mathit{vp} \,\&\, \mathit{vp}^1 \,\&\, \mathit{np} \mid w_{0,3}) \\ &= \Pr(\mathit{vp} \mid \mathit{vp}^1 \,\&\, \mathit{np} \,\&\, w_{0,3}) \\ &\quad \cdot \Pr(\mathit{vp}^1 \,\&\, \mathit{np} \mid w_{0,3}) \end{aligned} \tag{2}$$

The first factor is the likelihood that vp^1 and np, if they are in fact correct, will combine to make the bigger tree $\mathit{vp} \in \mathit{vp}^1 \otimes \mathit{np}$. We approximate it empirically, as discussed in the next section.

As for the second factor, the parser has already found it! It appeared as one of the summands in (1), which the parser used to find the LCP of np. It decomposes as

$$\begin{aligned} &\Pr(\mathit{vp}^1 \,\&\, \mathit{np} \mid w_{0,3}) \\ &\quad= \Pr(\mathit{vp}^1 \,\&\, \mathit{np} \,\&\, \mathit{np}^1 \,\&\, n \mid w_{0,3}) \\ &\quad= \Pr(\mathit{np} \mid \mathit{vp}^1 \,\&\, \mathit{np}^1 \,\&\, n \,\&\, w_{0,3}) \\ &\qquad \cdot \Pr(\mathit{vp}^1 \,\&\, \mathit{np}^1 \,\&\, n \mid w_{0,3}) \end{aligned} \tag{3}$$

The situation is exactly as before. We estimate the first factor empirically, and we have already found the second as

$$\begin{aligned} &\Pr(\mathit{vp}^1 \,\&\, \mathit{np}^1 \,\&\, n \mid w_{0,3}) \\ &\quad= \Pr(\mathit{vp}^1 \,\&\, \mathit{np}^1 \,\&\, n \,\&\, \mathit{call} \mid w_{0,3}) \\ &\quad= \Pr(n \mid \mathit{vp}^1 \,\&\, \mathit{np}^1 \,\&\, \mathit{call} \,\&\, w_{0,3}) \\ &\qquad \cdot \Pr(\mathit{vp}^1 \,\&\, \mathit{np}^1 \,\&\, \mathit{call} \mid w_{0,3}) \end{aligned} \tag{4}$$

At this point the recursion bottoms out, since *call* cannot be decomposed further. To find the second factor we invoke Bayes' theorem:

$$\begin{aligned} &\Pr(vp^1 \,\&\, np^1 \,\&\, call \mid w_{0,3}) \qquad\qquad (5)\\ =\ & \Pr(vp^1 \,\&\, np^1 \,\&\, call \mid w_{0,2} \,\&\, call)\\ =\ & \Pr(vp^1 \,\&\, np^1 \mid w_{0,2} \,\&\, call)\\ =\ & \frac{\Pr(call \mid vp^1 \,\&\, np^1 \,\&\, w_{0,2}) \cdot \Pr(vp^1 \,\&\, np^1 \mid w_{0,2})}{\sum_X \Pr(call \mid X \,\&\, w_{0,2}) \cdot \Pr(X \mid w_{0,2})}\end{aligned}$$

The sum in the denominator is over all chains X, including $vp^1 \,\&\, np^1$, that compete with each other to explain the input $w_{0,2} =$ *"Place a"*. Note that for each X, the LCP of $X \,\&\, call$ will have the same denominator.

In both the numerator and the denominator, the first factor is again estimated from corpus statistics. And again, the second factor has already been computed. For example, $\Pr(vp^1 \,\&\, np^1 \mid w_{0,2})$ is a summand in the LCP for np^1. Note that thanks to the Bayesian procedure, this is indeed a *left*-context probability: it does not depend on the word "call," which falls to the right of np^1.

Corpus Statistics

The recursive LCP computation does nothing but multiply together some empirical numbers. Where do these numbers come from? How does one estimate a value like $\Pr(vp \mid vp^1 \,\&\, np \,\&\, w_{0,3})$?

The condition $w_{0,3}$ is redundant, since the words also appear as the leaves of the chain $vp^1 \,\&\, np$. So the expression simplifies to $\Pr(vp \mid vp^1 \,\&\, np)$. This is the probability that, if vp^1 and np are correct in an arbitrary sentence, vp is also correct. (Consistent alternatives to $vp =$ [*VP v np*] might include [*VP v np pp*] and [*VP v* [*NP np pp*]], where *pp* is some prepositional phrase.)

In theory, one could find this value directly from the bracketed corpus:

(a) In the 3 bracketed training sentences (say) where

- $A =$ [*VP* [*V "Place"*] +]$_{0,1}$ appears
- $B =$ [*NP* [*Det "a"*] [*N "call"*]]$_{1,3}$ appears

in what fraction does
[*VP* [*V "Place"*] [*NP* [*Det "a"*] [*N "call"*]]]$_{0,3}$
appear?

However, such a question is too specific to be practical: 3 sentences is uncomfortably close to 0. To ensure that our samples are large enough, we broaden our question. We might ask instead:

(b) Among the 250 training sentences with

- $A =$ [*VP* [*V* . . .] +]$_{i,j}$
- $B =$ [*NP* . . .]$_{j,k}$ (some $i < j < k$)

in what fraction is B the second child of A?

Alternatively, we might take a more semantic approach and ask

(c) Among the 20 training sentences with

- $A =$ a subtree from i to j
- $B =$ a subtree from j to k
- A has semantic interpretation $A' =$ (*v-PLACE* . . .)
- B has semantic interpretation $B' =$ (*n-CALL* . . .)

in what fraction does B' fill the *OBJECT* role of A'?

Questions (b) and (c) both consider more sentences than (a). (b) considers a wide selection of sentences like "The operator activates CFA . . . ." (c) confines itself to sentences like "The operator places two priority calls . . . ."

As a practical matter, an estimate of $\Pr(vp \mid vp^1 \,\&\, np)$ should probably consider some syntactic *and* some semantic information about vp^1 and np. Our current approach is essentially to combine questions (b) and (c). That is, we focus our attention on corpus sentences that satisfy the conditions of both questions simultaneously.

Limiting Chain Length

The previous sections refer to many long chains of hypotheses:

$\Pr(vp^1 \,\&\, np^1 \,\&\, call \mid w_{0,3})$
$\Pr(n \mid vp^1 \,\&\, np^1 \,\&\, w_{0,3})$

In point of fact, every chain we have built extends back to the start of the sentence. But this is unacceptable: it means that parsing time and space grow exponentially with the length of the sentence.

Research in probabilistic parsing often avoids this problem by assuming a stochastic context-free language (SCFG). In this case,

$\Pr(vp^1 \,\&\, np^1 \,\&\, call \mid w_{0,3})$
$\quad = \Pr(vp^1 \mid w_{0,1}) \cdot \Pr(np^1 \mid w_{1,2}) \cdot \Pr(call \mid w_{2,3})$

and

$\Pr(n \mid vp^1 \,\&\, np^1 \,\&\, w_{0,3}) = \Pr(n \mid w_{2,3})$.

This assumption would greatly condense our computation and our statistical database. Unfortunately, for natural languages SCFGs are a poor model. Following the derivation $S \Rightarrow^*$ *John solved the* N, the probability of applying rule $\{N \rightarrow$ *fish*$\}$ is approximately zero—though it may be quite likely in other contexts.

But one may limit the length of chains less drastically, by making weaker independence assumptions. With appropriate modifications to the formulas, it is possible to arrange that chain length never exceeds some constant $L \geq 1$. Setting $L = 1$ yields the context-free case.

Our parser refines this idea one step further: within the bound L, chain length varies dynamically. For instance, suppose $L = 3$. In the parse tree

[*VP* [*V "Place"*]
    [*NP* [*Det "a"*] [*Adj "priority"*] [*N "call"*]]],

we do compute an LCP for the entire chain $vp^1 \,\&\, np^2 \,\&\, n$, but the chain $vp^1 \,\&\, np^1 \,\&\, adj$ is considered "too long." Why? The parser sees that *adj* will be buried two levels deep in both the syntactic and semantic trees for *vp*. It concludes that

$$\Pr(adj \mid vp^1 \,\&\, np^1) \approx \Pr(adj \mid np^1),$$

and uses this heuristic assumption to save work in several places.

In general we may speak of *high-focus* and *low-focus* parsing. The *focus* achieved during a parse is defined as the ratio of useful work to total work. Thus a high-focus parser is one that prunes aggressively: it does not allow incorrect hypotheses to proliferate. A completely deterministic parser would have 100% focus. Non-probabilistic parsers typically have low focus when applied to natural languages.

Our parsing algorithm allows us to choose between high- and low-focus strategies. To achieve high focus, we need a high value of ϵ (aggressive pruning) and accurate probabilities. This will prevent a combinatorial explosion. To the extent that accurate probabilities require long chains and complicated formulas, the useful work of the parser will take more time. On the other hand, a greater proportion of the work will be useful.

In practice, we find that even $L = 2$ yields good focus in the KITSS domain. Although the $L = 2$ probabilities undoubtedly sacrifice some degree of accuracy, they still allow us to prune most incorrect hypotheses. Related issues of modularity and local nondeterminism are also discussed in the psycholinguistic literature (e.g., [Tanenhaus et al 1985]).

Incomplete Knowledge

Natural language systems tend to be plagued by the potential open-endedness of the knowledge required for understanding. Incompleteness is a problem even for fully hand-coded systems. The correlate for statistical schemes is the problem of undertraining. In our task, for example, we do not have a large sample of syntactic and semantic bracketings. To compensate, the parser utilizes hand-coded knowledge as well as statistical knowledge. The hand-coded knowledge expresses general knowledge about linguistic subtheories such as part of speech rules or coordination.

As an example, consider the problem of assigning parts of speech to novel words. Several sources of knowledge may suggest the correct part of speech class—e.g., the left-context of the the novel word, the relative openness or closedness of the classes, morphological evidence from affixation, and orthographic conventions such as capitalization. The parser combines the various forms of evidence (using a simplified representation of the probability density functions) to assign a priori probabilities to novel part of speech rules. Using this technique, the parser takes a novel sentence such as "XX YY goes ZZ," and derives syntactic and semantic representations analogous to "Station B1 goes offhook."

The current system invokes these *ad hoc* knowledge sources when it fails to find suitable alternatives using its empirical knowledge. Clearly, there could be a more continuous strategy. Eventually, we would like to see very general linguistic principles (e.g., metarules and $\bar{X}$ theory [Jackendoff 1977]) and powerful informants such as world knowledge provide a priori biases that would allow the parser to guess genuinely novel structures. Ultimately, this knowledge may serve as an inductive bias for a completely bootstrapping, learning version of the system.

Results and Summary

The KITSS system is implemented in Franz Common Lisp running on Sun workstations. The system as a whole has now produced code for more than 70 complete test cases containing hundreds of sentences. The parsing component was trained and evaluated on a bracketed corpus of 429 sentences from 40 test cases. The bracketings use part of speech tags from the Brown Corpus [Francis and Kucera 1982] and traditional phrase structure labels (S, NP, etc.). Because adjunction rules such as $VP \rightarrow VP\ PP$ are common, the trees tend to be deep (9 levels on average). The bracketed 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. There are a total of 13,262 constituent decisions represented in the corpus.

We have studied the accuracy of the parser in using its statistics to reparse the training sentences correctly. Generally, we run the experiments using an adaptive beam schedule; progressively wider beam widths are tried (e.g., 10^{-1}, 10^{-2}, and 10^{-3}) until a parse is obtained. (By comparison, approximately 85% more hypotheses are generated if the system is run with a fixed beam of 10^{-3}.) For 62% of the sentences some parse was first found at $\epsilon = 10^{-1}$, for 32% at 10^{-2}, and for 6% at 10^{-3}. For only one sentence was no parse found within $\epsilon = 10^{-3}$. For the other 428 sentences, the parser always recovered the correct parse, and correctly identified it in 423 cases. In the 428 top-ranked parse trees, 99.92% of the bracketing decisions were correct (less than one error in 1000). Significantly, the parser produced only 1.02 parses per sentence. Of the hypotheses that survived pruning and were added to the chart, 68% actually appeared as parts of the final parse.

We have also done a limited number of randomized split-corpus studies to evaluate the parser's generalization ability. After several hundred sentences in the telephony domain, however, the vocabulary continues to grow and new structural rules are also occasionally needed. We have conducted a small-scale test that ensured that the test sentences were novel but grammatical under the known rules. With 211 sentences for training and 147 for testing, parses were found for 77% of the test sentences: of these, the top-ranked parse was correct 90% of the time, and 99.3% of the bracketing decisions were correct. Using 256 sentences for training and 102 sentences for testing, the parser performed perfectly on the test set.

In conclusion, we believe that the KITSS application demonstrates that it is possible to create a robust natural language processing system that utilizes both distributional knowledge and general linguistic knowledge.

Acknowledgements

Other major contributors to the KITSS system include Van Kelly, Uwe Nonnenmann, Bob Hall, John Eddy, and Lori Alperin Resnick.

Appendix A: The KITSS Parsing Algorithm

Algorithm 1. PARSE(w):

```
(* create a $(|w|+1) \times (|w|+1)$ chart $\mathbf{t} = (t_{i,j})$: *)
    $t_{0,0} := \{[S\,+]\}$;
    **for** j := 1 **to** $|w|$ **do**
        $R_1 := t_{j-1,j} := \{a_j\}$; $R_2 := \emptyset$;
        **while** $R_1 \neq \emptyset$
            $R :=$ PRUNE(($\mathbf{t} \otimes R_1) \cup$
                    (PREDICT($R_1$) $\otimes R_1) \cup R_2$);
            $R_1 := R_2 := \emptyset$;
            **for** $h_{i,j} \in R$ **do**
                $t_{i,j} := t_{i,j} \cup \{h_{i,j}\}$;
                **if** $h_{i,j}$ is complete
                    **then** $R_1 := R_1 \cup \{h_{i,j}\}$
                    **else** $R_2 := R_2 \cup \{h_{i,j}\}$
            **endfor**
        **endwhile**
    **endfor**;
    **return** { all complete $S$-hypotheses in $t_{0,|w|}$ }
```

Subroutine PREDICT(R):

```
    **return** $\{[A\,+] \mid A \rightarrow B_1 \ldots B_m$ is in $P$, some
            complete $B_1$-hypothesis is in $R$, and $A \neq S\}$
```

Subroutine PRUNE(R):

```
(* only likely rules are kept *)
    $R' := \emptyset$;
    $threshold := \epsilon \cdot \max_{h_{i,j} \in R} \Pr(h_{i,j} \mid w_{0,j})$;
    **for** $h_{i,j}$ **in** $R$ **do**
        **if** $\Pr(h_{i,j} \mid w_{0,j}) \geq threshold$
            **then** $R' := R' \cup \{h_{i,j}\}$
    **endfor**;
    **return** $R'$
```

References

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

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

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

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

[Jackendoff 1977] Jackendoff, R. 1977. *$\bar{X}$ Syntax: A Study of Phrase Structure.* Cambridge, MA.: MIT Press.

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

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

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

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