Reliability

Steven Abney

Universität Tübingen

The usual approach to stochastic parsing is to 'reverse' a model of stochastic generation. For example, a stochastic context-free grammar (SCFG) G defines a stochastic process for generating sentences. A stochastic choice is made at each derivation step, and the probability $P(d)$ of the derivation is the joint probability of the stochastic choices made. $P(d, x)$, the probability of producing derivation d and in the process deriving sentence x, is $P(d)$ if d is in fact a derivation of x, and 0 otherwise. $P(x)$ is $\sum_d P(d, x)$. To parse a sentence x, we choose that derivation d for which the conditional probability $P(d|x)$ of d given the sentence x is at a maximum, where $P(d|x)$ is defined as $P(d, x)/P(x)$.

However, there is reason to believe that no SCFG provides an acceptable approximation of $P(d|x)$ for English. The basic problem is that SCFG's with reasonable coverage of English typically assign significant probability to *many* parses for each sentence. As a result, even if an SCFG G is successful in the sense that the most-likely parse according to G is almost always the correct parse, the most-likely parse will nonetheless have a probability significantly less than one. In fact, it is probably a safe bet that, for SCFG's for English, the average probability for the most-likely parse is a good deal less than 1/2.

Since the SCFG parsing method simply chooses the most-likely parse as the correct parse, it might seem that only the relative likelihood of parses matters, not their absolute likelihood. But absolute likelihood does matter, for the following reason. If we care about which parse tree was used to generate a sentence (as we obviously do if we are interested in parsing), then we must consider English as a set of parse-trees, not terminal strings. $P(d, x)$ is our model of the probability of the speaker (writer) generating x with the parse-tree d. If the hearer (reader) always chooses the parse-tree d that maximizes $P(d|x)$, it follows that *the hearer chooses a different parse-tree from the one the speaker intended* with probability $1 - \max_d P(d|x)$. In a word, if p_{max} is the average probability of the most-likely parse, the misparse rate predicted by the model is $1 - p_{max}$. If the probability of the most-likely parse is only 1/2, we predict that people will misparse every other sentence they hear!

Misparses are generally considered quite rare. Certainly if genuine ambiguities were common, it would be impossible (or at least useless) to build treebanks in which a unique parse is assigned to each sentence. Mitch Marcus reports (p.c.) that in his experience in constructing the Penn Treebank, genuine ambiguities have been virtually nonexistent. A rate of 1 in 100 seems an overestimate. But even a rate of 1 in 10 would require a p_{max} of .90 on average, which is probably unachievable with SCFG's.

It may seem that the problem is just an issue of practicality: that SCFG's with appropriately low misparse rates do exist, even if we have not yet constructed one. But I think that unlikely.

First, we can show that the parsing problem for English *may* be such that no SCFG can solve it. To do so, we need some terminology. Define a *structured language* as a (possibly infinite) set of parse-trees. A treebank like the Penn Treebank (Marcus et al. 1993) is a sample of English as a structured language. A structured language D defines a unique (unstructured) language $L(D)$, being the union of the yields of parse-trees in D. There is also a unique context-free grammar $G(D)$ implicit in D: rule $X \to \alpha$ is in $G(D)$ iff X is expanded to α in some parse-tree in D. Conversely, the structured language $D(G)$ is the set of all parse-trees generated by G. A stochastic structured language is a structured language D and a probability distribution over D. For a stochastic structured language, $G(D)$ is an SCFG, where $X \to \alpha$ is in $G(D)$ iff X is expanded to α in some parse-tree in D, and the probability of $X \to \alpha$ is the relative frequency of expanding X to α in parse-trees of D, among all possible expansions of X in parse-trees of D. Conversely, $D(G)$ is the stochastic structured language that contains a parse-tree d with probability $P(d)$ iff G generates d with probability $P(d)$.

In these terms, what we can show is that structured languages D exist such that no context-free grammar generates D, and a fortiori, stochastic structured languages D exist such that no SCFG generates D.

Proof. Consider a structured language D. If any CFG generates D, it must be $G(D)$: any CFG G' that contains non-useless rules not in $G(D)$ ipso facto generates parse-trees not in D, and any CFG G' that omits rules in $G(D)$ ipso facto fails to generate some parse-trees in D. Now consider the structured language $D_0 = \{[_S a], [_S a[_S a]]\}$ with $G(D_0) = S \to aS|a$. $D(G(D_0)) \neq D_0$; hence no CFG generates D_0. QED.

(Note that the string language of D_0 is generated by a CFG, but not D_0. Note also that D_0 is the homomorphic image of the structured language of a CFG, e.g. $S \to a|aA, A \to a$. But D_0 itself is not generated by a CFG.)

Let us consider a more interesting example. Lexical analyzers for compilers typically consist of a set of regular-expression definitions, one for each category of token. A 'parse' for a sentence is a partitioning of the sentence into tokens. In general, there are many legal partitionings for a sentence, given a regular-expression grammar for tokens. The usual rule for choosing among them (e.g., Lesk & Schmidt 1979, Aho & Ullman 1972) is the longest-match rule: at the start position, choose the regular-expression match that covers the most of the input; then repeat, taking the end position of that match as the new start position.

For recursive grammars, the obvious analogue to the longest-match rule is 'close the current phrase as late as possible'—i.e., precisely the *low attachment* or *late closure* parsing preference (Kimball 1973, Frazier 1979, Shieber 1983), which accounts for the garden path status of examples like:

> The emergency crews really hate is domestic violence
> While Mary was mending the sock fell off her lap (Frazier 1978)
> I need one pound bag of sugar (Marcus 1980)

To evaluate the role that the longest-match rule plays in more pedestrian

sentences, I performed an experiment. I wrote a program to convert Penn Treebank trees into *chunks*—the non-recursive 'cores' of major phrases (Abney 1991). Based on a small sample, I wrote a regular-expression grammar to describe the chunks. I wanted to determine if applying the longest-match rule to the chunk grammar effectively discriminates correct matches (those that correspond to Treebank chunks) from incorrect matches. A match was considered a longest match if it was not a proper substring of some other match. On a test sample, longest matches corresponded to Treebank chunks 82% of the time. The largest source of error was lack of coverage of my grammar—if the true chunk was not covered by the grammar, the largest fragments of it that *were* covered by the grammar ended up as longest matches. If we eliminate such cases, in order to evaluate the effectiveness of the longest-match preference per se, longest matches are correct 95% of the time.

The longest-match rule cannot be captured by an SCFG, in general. Consider the regular grammar $G_1 =$

$$S \to AS|A$$
$$A \to aa|a$$

The language generated is $a+$, and the grammar is highly ambiguous. However, the longest-match rule specifies a unique correct parse for each sentence, in which every A is expanded to aa, except the last A in an odd-length sentence. It is easy to define a process with $P(d|x) = 1$ if d is correct, and $P(d|x) = 0$ otherwise. But there is no way to assign probabilities to the expansions in G so as to create an SCFG with $P(d_{best}|x) = 1$. Consider the sentence aaa, with correct parse $d_1 = [_S[_A aa][_S[_A a]]]$. To have $P(d_1|aaa) = 1$, we must have $P(d_1) > 0$. Since all rules in G are used in d_1, every rule must have probability > 0. As a consequence, all wrong parses for aaa must also have probability > 0; ergo $P(d_1|aaa)$ must be less than one.

It should not be surprising that SCFG's have difficulty capturing the longest-match rule. The notion of longest match is cross-derivational, which is to say, context-sensitive. We cannot determine whether a noun phrase like "the emergency" is a longest match by examining its internal structure; it is a longest match only if there is no alternative (partial) parse with a longer NP, and that depends on how the sentence goes on after "the emergency".

This suggests that we at least make probabilities of expansions context-sensitive. But expressing the longest-match preference is still difficult if our model is essentially a model of generation. Instead of considering the probability of expanding a category X to α, let us consider the probability of being right if we reduce α to X. That is, we interpret rule $X \to \alpha$ as a pattern that says sequence α can be reduced to X. The *reliability* of the pattern is the probability that a sequence α in fact reduces to a phrase X in the speaker's intended parse for the sentence. Reliability can be easily estimated from a treebank by counting. A tree of category X is correct if all its subtrees are correct, and if the pattern $X \to \alpha$ that licenses the tree is correct. Hence the reliability $R(d)$ of a (partial)

parse-tree d is the product of the reliabilities of its children times the reliability of the pattern licensing d.

Patterns can be refined by conditionalizing their reliability on further properties of the children and the context; e.g., by splitting instances of pattern α into longest matches vs. non-longest matches. For example, in the experiment reported above, only 31% of matches are correct. However, splitting into longest match and non-longest match improves discrimination: 98% of non-longest matches are wrong, and 82% of longest matches are right (not adjusting for coverage errors).

It is worth noting the relation between the model sketched here and *tree automata*. A tree automaton (Sakakibara 1992) walks a tree bottom-up. Its state at each node is a function of the category of the nodes and the automaton's state at each of the child nodes. The tree is accepted iff the automaton is in a final state at the root node. In the model I have sketched, there are two states: an accepting state and a rejecting state. I have also generalized tree automata by making the transition function context-sensitive and stochastic. Instead of specifying a single next state, the transition function provides a probability distribution over next states.

Finally, whether the pattern-reliability model turns out empirically to be superior to parsing-by-generation models (including context-sensitive variants such as that being explored at IBM: Black et al. 1993), a decided advantage of the pattern-reliability model is that it provides us with a framework for approaching the question of what cues enable people to parse (apparently) nearly deterministically in a single left-to-right pass. If we can find reliable patterns for recognizing phrases based on at most limited right context, we will have a candidate explanation not only for the fact of human parsing, but also its time course.

References

Steven Abney (1991). Parsing by Chunks. In: Berwick, Abney, & Tenny (eds.), *Principle-Based Parsing.* Kluwer.

Al Aho & Jeffrey Ullman (1972). *The Theory of Parsing, Translation, and Compiling.* Prentice-Hall, Englewood Cliffs, NJ.

Ezra Black, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer, & Salim Roukos (1993). Towards History-Based Grammars: Using Richer Models for Probabilistic Parsing. *Proc. ACL*, 31-37.

Lyn Frazier (1978). On Comprehending Sentences: Syntactic Parsing Strategies. Doctoral dissertation, University of Connecticut.

John Kimball (1973). Seven principles of surface structure parsing in natural language. *Cognition* 2, 15-47.

Michael Lesk and E. Schmidt (1979). LEX—lexical analyzer generator. In *UNIX Programmer's Manual,* University of California, Berkeley.

Mitchell P. Marcus (1980). *A Theory of Syntactic Recognition for Natural Language.* MIT Press, Cambridge, MA.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19.2, 313-330.

Stuart Shieber (1983). Sentence disambiguation by a shift-reduce parsing technique. *IJCAI-83*, 699-703.

Yasubumi Sakakibara (1992). Efficient Learning of Context-Free Grammars from Positive Structural Examples. *Information and Computation* 97, 23-60.