SOME CHART-BASED TECHNIQUES FOR PARSING ILL-FORMED INPUT

Chris S. Mellish
Department of Artificial Intelligence,
University of Edinburgh,
80 South Bridge,
EDINBURGH EH1 1HN,
Scotland.

ABSTRACT

We argue for the usefulness of an active chart as the basis of a system that searches for the globally most plausible explanation of failure to syntactically parse a given input. We suggest semantics-free, grammar-independent techniques for parsing inputs displaying simple kinds of ill-formedness and discuss the search issues involved.

THE PROBLEM

Although the ultimate solution to the problem of processing ill-formed input must take into account semantic and pragmatic factors, nevertheless it is important to understand the limits of recovery strategies that are based entirely on syntax and which are independent of any particular grammar. The aim of this work is therefore to explore purely syntactic and grammar-independent techniques to enable a parser to recover from simple kinds of ill-formedness in textual inputs. Accordingly, we present a generalised parsing strategy based on an active chart which is capable of diagnosing simple errors (unknown/misspelled words, omitted words, extra noise words) in sentences (from languages described by context free phrase structure grammars without ε-productions). This strategy has the advantage that the recovery process can run after a standard (active chart) parser has terminated unsuccessfully, without causing existing work to be repeated or the original parser to be slowed down in any way, and that, unlike previous systems, it allows the full syntactic context to be exploited in the determination of a "best" parse for an ill-formed sentence.

EXPLOITING SYNTACTIC CONTEXT

Weischedel and Sondheimer (1983) present an approach to processing ill-formed input based on a modified ATN parser. The basic idea is, when an initial parse fails, to select the incomplete parsing path that consumes the longest initial portion of the input, apply a special rule to allow the blocked parse to continue, and then to iterate this process until a successful parse is generated. The result is a "hill-climbing" search for the "best" parse, relying at each point on the "longest path" heuristic. Unfortunately, sometimes this heuristic will yield several possible parses, for instance with the sentence:

The snow blocks ↑ te road

(no partial parse getting past the point shown) where the parser can fail expecting either a verb or a determiner. Moreover, sometimes the heuristic will cause the most "obvious" error to be missed:

He said that the snow the road ↑
The paper will ↑ the best news is the Times

where we might suspect that there is a missing verb and a misspelled "with" respectively. In all these cases, the "longest path" heuristic fails to indicate unambiguously the minimal change that would be necessary to make the whole input acceptable as a sentence. This is not surprising, as the left-right bias of an ATN parser allows the system to take no account of the right context of a possible problem element.

Weischedel and Sondheimer's use of the "longest path" heuristic is similar to the use of locally least-cost error recovery in Anderson and Backhouse's (1981) scheme for compilers. It seems to be generally accepted that any form of globally "minimum-distance" error correction will be too costly to implement (Aho and Ullman, 1977). Such work has, however, not considered heuristic approaches, such as the one we are developing.

Another feature of Weischedel and Sondheimer's system is the use of grammar-specific recovery rules ("meta-rules" in their terminology). The same is true of many other systems for dealing with ill-formed input (e.g. Carbonell and Hayes (1983), Jensen et al. (1983)). Although grammar-specific recovery rules are likely in the end always to be more powerful than grammar-independent rules, it does seem to be worth investigating how far one can get with rules that only depend on the grammar *formalism* used.

<Need S from 0 to 7>	(hypothesis)
<Need NP+VP from 0 to 7>	(by top-down rule)
<Need VP from 2 to 7>	(by fundamental rule with NP found bottom-up)
<Need VP+PP from 2 to 7>	(by top-down rule)
<Need PP from 4 to 7>	(by fundamental rule with VP found bottom-up)
<Need P+NP from 4 to 7>	(by top-down rule)
<Need P from 4 to 5>	(by fundamental rule with NP found bottom-up)

Figure 1: Focusing on an error.

In adapting an ATN parser to compare partial parses, Weischedel and Sondheimer have already introduced machinery to represent several alternative partial parses simultaneously. From this, it is a relatively small step to introduce a well-formed substring table, or even an active chart, which allows for a global assessment of the state of the parser. If the grammar formalism is also changed to a declarative formalism (e.g. CF-PSGs, DCGs (Pereira and Warren 1980), Patr-II (Shieber 1984)), then there is a possibility of constructing other partial parses that do not start at the beginning of the input. In this way, right context can play a role in the determination of the "best" parse.

WHAT A CHART PARSER LEAVES BEHIND

The information that an active chart parser leaves behind for consideration by a "post mortem" obviously depends on the parsing strategy used (Kay 1980, Gazdar and Mellish 1989). *Active edges* are particularly important from the point of view of diagnosing errors, as an unsatisfied active edge suggests a place where an input error may have occurred. So we might expect to combine violated expectations with found constituents to hypothesise complete parses. For simplicity, we assume here that the grammar is a simple CF-PSG, although there are obvious generalisations.

(Left-right) *top-down parsing* is guaranteed to create active edges for each kind of phrase that could continue a partial parse starting at the beginning of the input. On the other hand, *bottom-up parsing* (by which we mean left corner parsing without top-down filtering) is guaranteed to find all complete constituents of every possible parse. In addition, whenever a non-empty initial segment of a rule RHS has been found, the parser will create active edges for the kind of phrase predicted to occur after this segment. Top-down parsing will always create an edge for a phrase that is needed for a parse, and so it will always indicate by the presence of an unsatisfied active edge the first error point, if there is one. If a subsequent error is present, top-down parsing will not always create an active edge corresponding to it, because the second may occur within a constituent that will not be predicted until the first error is corrected. Similarly, right-to-left top-down parsing will always indicate the last error point, and a combination of the two will find the first and last, but not necessarily any error points in between. On the other hand, bottom-up parsing will only create an active edge for each error point that comes immediately after a sequence of phrases corresponding to an initial segment of the RHS of a grammar rule. Moreover, it will not necessarily refine its predictions to the most detailed level (e.g. having found an NP, it may predict the existence of a following VP, but not the existence of types of phrases that can start a VP). Weischedel and Sondheimer's approach can be seen as an incremental top-down parsing, where at each stage the rightmost unsatisfied active edge is artificially allowed to be satisfied in some way. As we have seen, there is no guarantee that this sort of hill-climbing will find the "best" solution for multiple errors, or even for single errors. How can we combine bottom-up and top-down parsing for a more effective solution?

FOCUSING ON AN ERROR

Our basic strategy is to run a bottom-up parser over the input and then, if this fails to find a complete parse, to run a modified top-down parser over the resulting chart to hypothesise possible complete parses. The modified top-down parser attempts to find the minimal errors that, when taken account of, enable a complete parse to be constructed. Imagine that a bottom-up parser has already run over the input "the gardener collects manure if the autumn". Then Figure 1 shows (informally) how a top-down parser might focus on a possible error. To implement this kind of reasoning, we need a top-down parsing rule that knows how to refine a set of *global* needs and a

fundamental rule that is able to incorporate found constituents from either direction. When we may encounter multiple errors, however, we need to express multiple needs (e.g. <Need N from 3 to 4 and PP from 8 to 10>). We also need to have a fundamental rule that can absorb found phrases from anywhere in a relevant portion of the chart (e.g. given a rule "NP → Det Adj N" and a sequence "as marvellous sihgt", we need to be able to hypothesise that "as" should be a Det and "sihgt" a N). To save repeating work, we need a version of the top-down rule that stops when it reaches an appropriate category that has already been found bottom-up. Finally, we need to handle both "anchored" and "unanchored" needs. In an anchored need (e.g. <Need NP from 0 to 4>) we know the beginning and end of the portion of the chart within which the search is to take place. In looking for a NP VP sequence in "the happy blageon smupled the bait", however, we can't initially find a complete (initial) NP or (final) VP and hence don't know where in the chart these phrases meet. We express this by <Need NP from 0 to *, VP from * to 6>, the symbol "*" denoting a position in the chart that remains to be determined.

GENERALISED TOP-DOWN PARSING

If we adopt a chart parsing strategy with only edges that carry information about global needs, there will be considerable duplicated effort. For instance, the further refinement of the two edges:

<Need NP from 0 to 3 and V from 9 to 10>
<Need NP from 0 to 3 and Adj from 10 to 11>

can lead to any analysis of possible NPs between 0 and 3 being done twice. Restricting the possible format of edges in this way would be similar to allowing the "functional composition rule" (Steedman 1987) in standard chart parsing, and in general this is not done for efficiency reasons. Instead, we need to produce a single edge that is "in charge" of the computation looking for NPs between 0 and 3. When possible NPs are then found, these then need to be combined with the original edges by an appropriate form of the fundamental rule. We are thus led to the following form for a *generalised edge* in our chart parser:

<C from S to E needs
cs_1 from s_1 to e_1,
cs_2 from s_2 to e_2,
...
cs_n from s_n to e_n>

where C is a category, the cs_i are lists of categories (which we will show inside square brackets), S, E, the s_i and the e_i are positions in the chart (or the special symbol "*"). The presence of an edge of this kind in the chart indicates that the parser is attempting to find a phrase of category C covering the portion of the chart from S to E, but that in order to succeed it must still satisfy *all* the needs listed. Each need specifies a sequence of categories cs_i that must be found contiguously to occupy the portion of the chart extending from s_i to e_i.

Now that the format of the edges is defined, we can be precise about the parsing rules used. Our modified chart parsing rules are shown in Figure 2. The modified *top-down rule* allows us to refine a need into a more precise one, using a rule of the grammar (the extra conditions on the rule prevent further refinement where a phrase of a given category has already been found within the precise part of the chart being considered). The modified *fundamental rule* allows a need to be satisfied by an edge that is completely satisfied (i.e. an inactive edge, in the standard terminology). A new rule, the *simplification rule*, is now required to do the relevant housekeeping when one of an edge's needs has been completely satisfied. One way that these rules could run would be as follows. The chart starts off with the inactive edges left by bottom-up parsing, together with a single "seed" edge for the top-down phase <GOAL from 0 to n needs [S] from 0 to n>, where n is the final position in the chart. At any point the fundamental rule is run as much as possible. When we can proceed no further, the first need is refined by the top-down rule (hopefully search now being anchored). The fundamental rule may well again apply, taking account of smaller phrases that have already been found. When this has run, the top-down rule may then further refine the system's expectations about the parts of the phrase that cannot be found. And so on. This is just the kind of "focusing" that we discussed in the last section. If an edge expresses needs in several separate places, the first will eventually get resolved, the simplification rule will then apply and the rest of the needs will then be worked on.

For this all to make sense, we must assume that all hypothesised needs can eventually be resolved (otherwise the rules do not suffice for more than one error to be narrowed down). We can ensure this by introducing special rules for recognising the most primitive kinds of errors. The results of these rules must obviously be scored in some way, so that errors are not wildly hypothesised in all sorts of places.

Top-down rule:
<C from S to E needs [c_1...cs_1] from s_1 to e_1, cs_2 from s_2 to e_2, ... cs_n from s_n to e_n>
$c_1 \rightarrow$ RHS (in the grammar)

<c_1 from s_1 to e needs RHS from s_1 to e>
where e = if cs_1 is not empty or e_1 = * then * else e_1
(e_1 = * or cs_1 is non-empty or there is no category c_1 from s_1 to e_1)

Fundamental rule:
<C from S to E needs [...cs_{11} c_1 ...cs_{12}] from s_1 to e_1, cs_2 ...>
<c_1 from S_1 to E_1 needs <nothing>>

<C from S to E needs cs_{11} from s_1 to S_1, cs_{12} from E_1 to e_1, cs_2 ...>
($s_1 \leq S_1$, e_1 = * or $E_1 \leq e_1$)

Simplification rule:
<C from S to E needs [] from s to s, cs_2 from s_2 to e_2, ... cs_n from s_n to e_n>

<C from S to E needs cs_2 from s_2 to e_2, ... cs_n from s_n to e_n>

Garbage rule:
<C from S to E needs [] from s_1 to e_1, cs_2 from s_2 to e_2, ... cs_n from s_n to e_n>

<C from S to E needs cs_2 from s_2 to e_2, ... cs_n from s_n to e_n>
($s_1 \neq e_1$)

Empty category rule:
<C from S to E needs [c_1...cs_1] from s to s, cs_2 from s_2 to e_2, ... cs_n from s_n to e_n>

<C from S to E needs cs_2 from s_2 to e_2, ... cs_n from s_n to e_n>

Unknown word rule:
<C from S to E needs [c_1...cs_1] from s_1 to e_1, cs_2 from s_2 to e_2, ... cs_n from s_n to e_n>

<C from S to E needs cs_1 from s_1+1 to e_1, cs_2 from s_2 to e_2, ... cs_n from s_n to e_n>
(c_1 a lexical category, s_1 < the end of the chart and
the word at s_1 not of category c_1).

Figure 2: Generalised Top-down Parsing Rules

SEARCH CONTROL AND EVALUATION FUNCTIONS

Even without the extra rules for recognising primitive errors, we have now introduced a large parsing search space. For instance, the new fundamental rule means that top-down processing can take place in many different parts of the chart. Chart parsers already use the notion of an *agenda*, in which possible additions to the chart are given priority, and so we have sought to make use of this in organising a heuristic search for the "best" possible parse. We have considered a number of parameters for deciding which edges should have priority:

MDE (mode of formation) We prefer edges that arise from the fundamental rule to those that arise from the top-down rule; we disprefer edges that arise from unanchored applications of the top-down rule.

PSF (penalty so far) Edges resulting from the garbage, empty category and unknown word rules are given penalty scores. PSF counts the penalties that have been accumulated so far in an edge.

PB (best penalty) This is an estimate of the best possible penalty that this edge, when complete, could have. This score can use the PSF, together with information about the parts of the chart covered - for

instance, the number of words in these parts which do not have lexical entries.

GUS (the maximum number of words that have been used so far in a partial parse using this edge) We prefer edges that lead to parses accounting for more words of the input.

PBG (the best possible penalty for any complete hypothesis involving this edge). This is a *shortfall* score in the sense of Woods (1982).

UBG (the best possible number of words that could be used in any complete hypothesis containing this edge).

In our implementation, each rule calculates each of these scores for the new edge from those of the contributing edges. We have experimented with a number of ways of using these scores in comparing two possible edges to be added to the chart. At present, the most promising approach seems to be to compare in turn the scores for PBG, MDE, UBG, GUS, PSF and PB. As soon as a difference in scores is encountered, the edge that wins on this account is chosen as the preferred one. Putting PBG first in this sequence ensures that the first solution found will be a solution with a minimal penalty score.

The rules for computing scores need to make estimates about the possible penalty scores that might arise from attempting to find given types of phrases in given parts of the chart. We use a number of heuristics to compute these. For instance, the presence of a word not appearing in the lexicon means that every parse covering that word must have a non-zero penalty score. In general, an attempt to find an instance of a given category in a given portion of the chart must produce a penalty score if the bottom-up parsing phase has not yielded an inactive edge of the correct kind within that portion. Finally, the fact that the grammar is assumed to have no ε-productions means that an attempt to find a long sequence of categories in a short piece of chart is doomed to produce a penalty score; similarly a sequence of lexical categories cannot be found without penalty in a portion of chart that is too long.

Some of the above scoring parameters score an edge according what sorts of parses it could contribute to, not just according to how internally plausible it seems. This is desirable, as we wish the construction of globally most plausible solutions to drive the parsing. On the other hand, it introduces a number of problems for chart organisation. As the same edge (apart from its score) may be generated in different ways, we may end up with multiple possible scores for it. It would make sense at each point to consider the best of the possible scores associated with an edge to be the current score. In this way we would not have to repeat work for every differently scored version of an edge. But consider the following scenario:

> Edge A is added to the chart. Later edge B is spawned using A and is placed in the agenda. Subsequently A's score increases because it is derived in a new and better way. This should affect B's score (and hence B's position on the agenda).

If the score of an edge increases then the scores of edges on the agenda which were spawned from it should also increase. To cope with this sort of problem, we need some sort of dependency analysis, a mechanism for the propagation of changes and an easily resorted agenda. We have not addressed these problems so far - our current implementation treats the score as an integral part of an edge and suffers from the resulting duplication problem.

PRELIMINARY EXPERIMENTS

To see whether the ideas of this paper make sense in practice, we have performed some very preliminary experiments, with an inefficient implementation of the chart parser and a small CF-PSG (84 rules and 34 word lexicon, 18 of whose entries indicate category ambiguity) for a fragment of English. We generated random sentences (30 of each length considered) from the grammar and then introduced random occurrences of specific types of errors into these sentences. The errors considered were none (i.e. leaving the correct sentence as it was), deleting a word, adding a word (either a completely unknown word or a word with an entry in the lexicon) and substituting a completely unknown word for one word of the sentence. For each length of original sentence, the results were averaged over the 30 sentences randomly generated. We collected the following statistics (see Table 1 for the results):

BU cycles - the number of cycles taken (see below) to exhaust the chart in the initial (standard) bottom-up parsing phase.

#Solns - the number of different "solutions" found. A "solution" was deemed to be a description of a possible set of errors which has a minimal penalty score and if corrected would enable a complete parse to be constructed. Possible errors were adding an extra word, deleting a word and substituting a word for an instance of a given lexical category.

Table 1: Preliminary experimental results

Error	Length of original	BU cycles	#Solns	First	Last	TD cycles
None	3	31	1	0	0	0
	6	69	1	0	0	0
	9	135	1	0	0	0
	12	198	1	0	0	0
Delete one word	3	17	5	14	39	50
	6	50	5	18	73	114
	9	105	6	27	137	350
	12	155	7	33	315	1002
Add unknown word	3	29	1	9	17	65
	6	60	2	24	36	135
	9	105	2	39	83	526
	12	156	3	132	289	1922
Add known word	3	37	3	29	51	88
	6	72	3	43	88	216
	9	137	3	58	124	568
	12	170	5	99	325	1775
Subst unknown word	3	17	2	17	28	46
	6	49	2	23	35	105
	9	96	2	38	56	300
	12	150	3	42	109	1162

The penalty associated with a given set of errors was the number of errors in the set.

First - the number of cycles of generalised top-down parsing required to find the first solution.

Last - the number of cycles of generalised top-down parsing required to find the last solution.

TD cycles - the number of cycles of generalised top-down parsing required to exhaust all possibilities of sets of errors with the same penalty as the first solution found.

It was important to have an implementation-independent measure of the amount of work done by the parser, and for this we used the concept of a "cycle" of the chart parser. A "cycle" in this context represents the activity of the parser in removing one item from the agenda, adding the relevant edge to the chart and adding to the agenda any new edges that are suggested by the rules as a result of the new addition. For instance, in conventional top-down chart parsing a cycle might consist of removing the edge <S from 0 to 6 needs [NP VP] from 0 to 6> from the front of the agenda, adding this to the chart and then adding new edges to the agenda, as follows. First of all, for each edge of the form <NP from 0 to α needs []> in the chart the fundamental rule determines that <S from 0 to 6 needs [VP] from α to 6> should be added. Secondly, for each rule $NP \rightarrow \gamma$ in the grammar the top-down rule determines that <NP from 0 to * needs γ from 0 to *> should be added. With generalised top-down parsing, there are more rules to be considered, but the idea is the same. Actually, for the top-down rule our implementation schedules a whole collection of single additions ("apply the top down rule to edge α") as a single item on the agenda. When such a request reaches the front of the queue, the actual new edges are then computed and themselves added to the agenda. The result of this strategy is to make the agenda smaller but more structured, at the cost of some extra cycles.

EVALUATION AND FUTURE WORK

The preliminary results show that, for small sentences and only one error, enumerating all the possible minimum-penalty errors takes no worse than 10 times as long as parsing the correct sentences. Finding the first minimal-penalty error can also be quite fast. There is, however, a great variability between the types of error. Errors involving completely unknown words can be diagnosed reasonably

quickly because the presence of an unknown word allows the estimation of penalty scores to be quite accurate (the system still has to work out whether the word can be an addition and for what categories it can substitute for an instance of, however). We have not yet considered multiple errors in a sentence, and we can expect the behaviour to worsten dramatically as the number of errors increases. Although Table 1 does not show this, there is also a great deal of variability between sentences of the same length with the same kind of introduced error. It is noticeable that errors towards the end of a sentence are harder to diagnose than those at the start. This reflects the left-right orientation of the parsing rules - an attempt to find phrases starting to the right of an error will have a PBG score at least one more than the estimated PB, whereas an attempt to find phrases in an open-ended portion of the chart starting before an error may have a PBG score the same as the PB (as the error may occur within the phrases to be found). Thus more parsing attempts will be relegated to the lower parts of the agenda in the first case than in the second.

One disturbing fact about the statistics is that the number of minimal-penalty solutions may be quite large. For instance, the ill-formed sentence:

who has John seen on that had

was formed by adding the extra word "had" to the sentence "who has John seen on that". Our parser found three other possible single errors to account for the sentence. The word "on" could have been an added word, the word "on" could have been a substitution for a complementiser and there could have been a missing NP after "on". This large number of solutions could be an artefact of our particular grammar and lexicon; certainly it is unclear how one should choose between possible solutions in a grammar-independent way. In a few cases, the introduction of a random error actually produced a grammatical sentence - this occurred, for instance, twice with sentences of length 5 given one random added word.

At this stage, we cannot claim that our experiments have done anything more than indicate a certain concreteness to the ideas and point to a number of unresolved problems. It remains to be seen how the performance will scale up for a realistic grammar and parser. There are a number of detailed issues to resolve before a really practical implementation of the above ideas can be produced. The indexing strategy of the chart needs to be altered to take into account the new parsing rules, and remaining problems of duplication of effort need to be addressed. For instance, the generalised version of the fundamental rule allows an active edge to combine with a set of inactive edges satisfying its needs in any order.

The scoring of errors is another area which should be better investigated. Where extra words are introduced accidentally into a text, in practice they are perhaps unlikely to be words that are already in the lexicon. Thus when we gave our system sentences with known words added, this may not have been a fair test. Perhaps the scoring system should prefer added words to be words outside the lexicon, substituted words to substitute for words in open categories, deleted words to be non-content words, and so on. Perhaps also the confidence of the system about possible substitutions could take into account whether a standard spelling corrector can rewrite the actual word to a known word of the hypothesised category. A more sophisticated error scoring strategy could improve the system's behaviour considerably for real examples (it might of course make less difference for random examples like the ones in our experiments).

Finally the behaviour of the approach with realistic grammars written in more expressive notations needs to be established. At present, we are investigating whether any of the current ideas can be used in conjunction with Allport's (1988) "interesting corner" parser.

ACKNOWLEDGEMENTS

This work was done in conjunction with the SERC-supported project GR/D/16130. I am currently supported by an SERC Advanced Fellowship.

REFERENCES

Aho, Alfred V. and Ullman, Jeffrey D. 1977 *Principles of Compiler Design*. Addison-Wesley.

Allport, David. 1988 The TICC: Parsing Interesting Text. In: Proceedings of the Second ACL Conference on Applied Natural Language Processing, Austin, Texas.

Anderson, S. O. and Backhouse, Roland C. 1981 Locally Least-Cost Error-Recovery in Earley's Algorithm. *ACM TOPLAS* 3(3): 318-347.

Carbonell, Jaime G. and Hayes, Philip J. 1983 Recovery Strategies for Parsing

Extragrammatical Language. *AJCL* 9(3-4): 123-146.

Gazdar, Gerald and Mellish, Chris. 1989 *Natural Language Processing in LISP - An Introduction to Computational Linguistics.* Addison-Wesley.

Jensen, Karen, Heidorn, George E., Miller, Lance A. and Ravin, Yael. 1983 Parse Fitting and Prose Fitting: Getting a Hold on Ill-Formedness. *AJCL* 9(3-4): 147-160.

Kay, Martin. 1980 Algorithm Schemata and Data Structures in Syntactic Processing. Research Report CSL-80-12, Xerox PARC.

Pereira, Fernando C. N. and Warren, David H. D. 1980 Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence* 13(3): 231-278.

Shieber, Stuart M. 1984 The Design of a Computer Language for Linguistic Information. In Proceedings of COLING-84, 362-366.

Steedman, Mark. 1987 Combinatory Grammars and Human Language Processing. In: Garfield, J., Ed., *Modularity in Knowledge Representation and Natural Language Processing.* Bradford Books/MIT Press.

Weischedel, Ralph M. and Sondheimer, Norman K. 1983 Meta-rules as a Basis for Processing Ill-Formed Input. *AJCL* 9(3-4): 161-177.

Woods, William A. 1982 Optimal Search Strategies for Speech Understanding Control. *Artificial Intelligence* 18(3): 295-326.