Evaluating Parsing Strategies Using Standardized Parse Files

Ralph Grishman and **Catherine Macleod** and **John Sterling**[*]
Computer Science Department
New York University
715 Broadway, 7th Floor
New York, New York 10003

Abstract

The availability of large files of manually-reviewed parse trees from the University of Pennsylvania "tree bank", along with a program for comparing system-generated parses against these "standard" parses, provides a new opportunity for evaluating different parsing strategies. We discuss some of the restructuring required to the output of our parser so that it could be meaningfully compared with these standard parses. We then describe several heuristics for improving parsing accuracy and coverage, such as closest attachment of modifiers, statistical grammars, and fitted parses, and present a quantitative evaluation of the improvements obtained with each strategy.

1 The Problem

The systematic improvement of parsing strategies requires some way of evaluating competing strategies. We need to understand their effect on both overall system performance and the ability to parse sentences correctly. In order to evaluate parser output, we will need a file of standard "correct" parses and a mechanism for comparing parser output with this standard. Preparing such a file is a quite time-consuming operation. In addition, if we would like our comparison of strategies to be meaningful to other researchers and extendible to other systems, we would like a standard and a metric which are, as much as possible, system independent.

One resource which is newly available to meet this need is the "tree bank" at the University of Pennsylvania. This bank includes a large number of parse trees which have been prepared manually following a fairly detailed standard. At first glance the comparison of parse trees among systems based on different theories would seem to be very difficult. However, based on an idea proposed by Ezra Black [Black *et al.*, 1991], a group organized by Phil Harrison has developed a metric and a program for performing just such a comparison. A preliminary assessment of this metric, based on a small sample of short sentences (50 13-word sentences), has been quite promising [Harrison *et al.*, 1991].

Our goal, in the work described here, was to extend this effort to larger samples and more complex sentences and to use the metric to compare various parsing heuristics which we have employed in our system. We describe briefly in the next sections the nature of the task and the text to which is was applied, the form of the tree bank, and the comparison metric. We consider some of the systematic mis-alignments between the standard and our output, and describe how we have reduced these. Finally, we describe some of our parsing strategies and see how they affect both the parse scores and the overall system performance.

2 The Application Task

The experiments described below were performed on a version of the PROTEUS system as prepared for MUC-3, the third Message Understanding Conference [Sundheim, 1991]. The task set before the participants at these Conferences is one of *information extraction*: taking free-text input on a particular subject matter, extracting specified types of information, and filling a data base with this information. For MUC-3, the texts consisted of short news reports about terrorist incidents; most are in typical newspaper style, with complex sentences, although transcripts of speeches and radio broadcasts are also included.

The PROTEUS system has five basic stages of processing: syntactic analysis, semantic analysis, reference resolution, discourse analysis, and data base creation. Syntactic analysis consists of parsing with a broad-coverage English grammar and dictionary, followed by syntactic regularization. The regularized parse is then given to the semantic analyzer, which produces a predicate-argument structure. The semantic analyzer also provides feedback to the parser as to whether the parse satisfies semantic (selectional) constraints. Thus correct data base creation is dependent on correct predicate-argument structures, which are in turn dependent on at least locally correct syntactic structures.

Because of the richness of the text, it is not possible to provide a complete set of semantic patterns. The semantic patterns included in the system are largely limited to those relevant for filling data base entries. For the terrorist domain, this includes patterns for all sorts of terrorist

[*]This paper is based upon work supported by the Defense Advanced Research Projects Agency under Grant N00014-90-J-1851 from the Office of Naval Research, and by the National Science Foundation under Grant IRI-89-02304.

incidents (attacks, bombings, ...), for the entities which can be involved in these incidents (as perpetrators, targets, instruments, ...), and for other structures which may affect truth value or certainty (claim, deny, allege, ...). We use a system of preference semantics which penalizes but does not exclude constructs not matching any of these semantic patterns [Grishman and Sterling, 1990]. Our parser does a best-first search for the analysis with the lowest penalty.

The information extraction task has the benefit of providing relatively clear measures of overall system performance. As part of MUC-3, a set of standard (correct) data base entries was prepared and a program was developed for scoring system responses against this standard. Roughly speaking, the scoring program computes three counts: Std, the number of data base fills in the standard (correct) data base; Sys, the number of fills generated by the system; and Cor, the number of correct fills generated by the system. System performance is then stated in terms of recall (= Cor / Std) and precision (= Cor / Sys).

3 The Tree Bank

The goal of the "Treebank" at the University of Pennsylvania is to construct a large data base of English annotated with detailed grammatical structure. Among the texts which they have annotated is a portion of the development corpus used for MUC-3; they have annotated 356 of the 1300 articles in that corpus.

Sentences are annotated in a semi-automated procedure in which sentences are first parsed automatically and these parses are then manually revised. The result is a file of labeled bracketings conforming in general terms to the syntactic structures of transformational generative grammar (X-bar theory). The bracketing is shown in considerable detail at the clause level; some of the details at the NP level are suppressed. The guidelines for the bracketing are given in [Santorini, 1991]. A sample sentence from the MUC-3 corpus, as bracketed for the Treebank, is shown in Figure 1.

4 The Evaluation Metric

The basic idea for the evaluation metric was developed by Ezra Black. It was then refined and tested at a workshop (and through extensive subsequent electronic communication) organized by Phil Harrison. This effort has involved computational linguists from eight sites who have applied this metric to compare — for a set of 50 short sentences — the Penn standard with the "ideal" parses their systems would generate for these sentences.

At first glance the parses produced by different systems may seem so different as to be incomparable; node labels in particular may be entirely different. The metric therefore eliminates node labels and only compares tree structures. Certain constituents are treated very differently by different systems; for example, auxiliaries are treated as main verbs by some systems but not others. Several classes of constituents, such as auxiliaries, pre-infinitival "to", "not", null elements, punctuation,

```
( (S (PP in (NP cartagena))
     (S (NP it)
        was
        (VP reported
            (SBAR
             (SBAR
              that
              (S (NP castellar)
                 (VP faced
                     (NP a
                         ``
                         revolutionary trial
                         ''
                         (PP by
                             (NP the ELN))))))
             and
             (SBAR that
                   (S (NP he)
                      was
                      (VP (VP found
                              (ADJP guilty))
                          and
                          (VP executed)))))))
  .)
```

Figure 1: *A sample sentence from the U. of Pennsylvania Tree Bank*

etc., are therefore deleted before trees are compared. After these eliminations, brackets with no elements inside, brackets around single words, and multiple brackets surrounding the same sequence of words are removed. Finally, these two bracketed sequences are compared.

We count the number of brackets in the standard output (Std), the total number in the system output (Sys), and the number of matching brackets (M). We then define — just as we did for system performance — measures of recall (= M / Std) and precision (= M / Sys). We also count the number of "crossings": the number of cases where a bracketed sequence from the standard overlaps a bracketed sequence from the system output, but neither sequence is properly contained in the other. Most automatic parsing systems generate considerably more detailed bracketing than that produced by the Treebank; in such cases precision would be reduced, but the recall and crossing count would still accurately reflect the correctness of the system-generated parses.

Phil Harrison and Steven Abney have developed and distributed a program which computes these various measures.

5 Improving Alignment

Even with all the parse tree simplifications described in the previous section, there can still be substantial systematic differences between the Treebank standard and system output. The NYU system is a particularly good (bad?) example in this regard, since it is based on Harris's Linguistic String Theory and the trees produced are therefore quite different in several respects from X-bar trees. The result is a uniform degradation of recall and precision.

To reduce these differences, we have implemented a simple tree transducer to restructure the output of our parser. This is implemented as a set of tree rewriting rules which are applied at all levels of the parse tree. Our primary goal has been to reduce crossings and increase recall by producing trees which correspond more closely to the standard; secondarily, we have also aimed to increase precision by eliminating levels of bracketing in our trees.

One of the principal differences is that our grammar divides the sentence (ASSERTION) into SUBJECT, VERB, and OBJECT, while the standard divides it into NP (subject) and VP (verb + object). We therefore include a rule to insert a VP node into our trees:

```
((ASSERTION (SA . ?sa1)
            (SUBJECT . ?subj)
            (SA . ?sa2)
            (VERB . ?verb)
            (SA . ?sa3)
            (OBJECT . ?obj)
            (SA . ?sa4))
->
(S (SA . ?sa1)
   (S (SUBJECT . ?subj)
      (SA . ?sa2)
      (VP (VERB . ?verb)
          (SA . ?sa3)
          (OBJECT . ?obj))
      (SA . ?sa4))))
```

This rule is complicated by the presence of SAs (sentence adjuncts), which may be empty. Depending on their position, they are placed in the restructured tree beneath VP, beneath S, or as a sister of the main S.

Differences in basic constituent structure have further ramifications for the scoping of coordinate conjunction. For example, "The terrorists attacked the village and murdered the mayor." will be analyzed as *NP VP and VP* in the standard, while the PROTEUS system will analyze it as *ASSERTION and ASSERTION*, with the *SUBJECT* of the second *ASSERTION* empty. We therefore include a rule to restructure this as *VP and VP*.

Rewrite rules are also required to handle differences in the treatment of right modifiers of the noun. We have a uniform structure for the NP, *left modifiers + head noun + right modifiers*. The standard, in contrast, uses rules of the form $NP \rightarrow NP\ VP$, so that a verbal right modifier is treated as a right sister of the structure consisting of the head and left modifiers. Finally, since the standard employs only minimal bracketing of left modifiers of the noun, we have included rules to eliminate most of that structure from our trees. All told, we currently have 26 rewriting rules.

Our set of restructuring rules is not complete. More brackets would need to be eliminated to match the level of bracketing used in the standard. Also, we treat certain phrases (such as "according to", "because of", and "as a consequence of") as idioms, and so bracket them as a single unit, whereas they are assigned a full phrase structure in the standard. To gauge how closely our restructured trees match the standard, we have taken a set of 13 sentences which appear to parse correctly with our system and scored them against the standard. Without the restructuring rules we obtained an average recall of 85.38%, an average precision of 76.55%, and an average of 0.15 crossings per sentence. Applying the restructuring rules improved these scores to an average recall of 94.62%, an average precision of 92.48%, and an average of 0.08 crossings per sentence.

6 Evaluation

For an evaluation of our different parsing heuristics, we took the first 33 of the 356 news reports from the MUC-3 corpus which had been bracketed by the University of Pennsylvania. Three of these were deleted because of differences in the way the sentence boundaries were assigned by our system and UPenn. This left 30 reports with 317 sentences for the evaluation runs.

Our syntactic analyzer uses an augmented-context-free grammar: a context-free core plus a set of constraints. Some of these are stated as absolute constraints: if the constraint is violated, the analysis is rejected. Others are stated as preferences: associated with each of these constraints is a penalty to be assessed if the constraint is violated. Penalties from different constraints are combined additively, and the analysis with the least penalty is selected as the final parse. The parser uses a best-first search in which, at each point, the hypothesis with the lowest penalty is pursued. Among the constraints which are realized as preferences in the system are: some grammatical constraints (including adverb position, count noun, and comma constraints), closest attachment of modifiers, statistical preference for productions, and all semantic constraints.

6.1 The base run

As a "base run", we excluded almost all preferential constraints. We left in penalties on two constructs which, while grammatical, lead to substantial extra computation — headless noun phrases and relative clauses without relative pronouns. We also left in the ability to relax (with penalty) a few grammatical constraints: constraints on adverb position, the constraint on count nouns requiring determiners, and the constraints on commas. Most sentences do not involve either of these constructs or violate any of these constraints, and so do not incur any penalty. In fact, most sentences get lots of parses under these conditions: on the average we got 60 parses per sentence (these sentences are an average of 22.5 words long).

Since the parsing algorithm requires in the worst case exponential time, we place some bound on the number of hypotheses (edges in the chart) which may be generated for each sentence; for the current evaluation, we used a limit of 33000. For some sentences, all hypotheses can be explored within this limit; for the others, parsing is terminated when the edge limit is reached. Each set of sentences may be further divided into those which obtain one or more parses and those which do not obtain any. Adding heuristics in the form of additional preferences may reduce the search space, so that some sentences which previously hit the edge limit without getting a parse will now get a parse. These preferences can thus improve recall and precision in two distinct ways: by selecting among parses produced by the base run in better than random fashion, and by shifting some sentences from the "no parse (edge limit)" column to the "parsed" column. In order to separate these two effects, we also compute the averages of crossings, recall, and precision over the subset of 206 sentences which parsed in the base run and did not reach the edge limit. Improvements in these numbers will show the value of a particular preference as a filter in selecting among parses, separate from its value in guiding the parsing process.

For the base run on the subset of 206 sentences, we computed two sets of statistics. For the first set, when a sentence has $N > 1$ parses, we weight each parse by $1/N$ in computing the average number of crossings and the total recall and precision. This is in some sense a fair evaluation of how well we would do if we were forced to pick a single parse for each sentence and we did so at random. The second set of statistics is computed by selecting (from the parses for one sentence) the parse with the minimal number of crossings and, among these the parse with the highest recall. This represents how well we could do — given the limitations of our base grammar — if we had an ideal filter to select among the parses generated.

6.2 Parsing Heuristics

We describe below briefly the various heuristics used in our experiments. The results computed over the entire corpus are summarized in Table 1; results over the 206 sentences which parsed in the base run without hitting the edge limit are summarized in Table 2. For sentences with $N > 1$ parses, we weight each parse by $1/N$ in computing the average number of crossings and total recall and precision (as we did for the first set of statistics for the base run, described in the previous paragraph). In addition, we have included in Table 2, line 1m, the second set of statistics for the base run, based on picking the best parse for each sentence.

6.2.1 Closest attachment

The simplest preference we tested was closest attachment of modifiers. We penalized each modifier by the number of words separating the modifier from the word it modifies, with some additional penalty for successive modifiers at the same level of the tree. This produced an improvement in performance on all measures (line 2 in both tables).

6.2.2 Statistical grammar

Using a sample of 260 sentence from the MUC-3 corpus (disjoint from the set used here for evaluation), we computed the probability of each production in our context-free grammar using an iterative unsupervised training scheme similar to the inside-outside algorithm [Fujisaki, 1984; Chitrao and Grishman, 1990; Chitrao, 1990]. We then used the logarithms of the probabilities as penalties in applying the productions during our analysis of the evaluation corpus. We used these statistical weights by themselves and in combination with closest attachment. Note that statistical weighting by itself is not particularly effective in focusing the search, since longer hypotheses almost invariably have higher penalties than shorter ones; as we have previously reported, its effectiveness is greatly increased when combined with a weighting scheme which prefers longer hypotheses. Statistical weighting by itself produced an improvement on all measures (line 3). Furthermore, the combination of statistical weighting and closest attachment (line 4) did better than either heuristic separately, except for a slight loss in average recall over the entire corpus.

6.2.3 Merging

In our base parser, alternative hypotheses are never merged. If X is a non-terminal symbol of our grammar, we may generate several separate edges in the chart representing X spanning words w_1 to w_2, each representing a different analysis. When merging is enabled, we retain only the highest-scoring analysis, so there will be only one edge for each non-terminal and span of words.[1] Merging, when added to the statistical grammar and closest attachment, had little effect on the performance of sentences which parsed in the base run (Table 2, line 5). However, it did substantially increase the number of sentences parsed (by reducing the number of hypotheses to be followed, it allowed sentences which had previously hit the edge limit to complete), and therefore increased the average recall over the entire corpus (and the number of crossings).

6.2.4 Junk

Although our grammar has moderately broad coverage, there are certainly still a fair number of grammatical structures which have not been included. In addition, there are frequent passages not conforming to "standard grammar", particularly in the reports containing verbatim transcripts of speeches. Recognizing that most of these sentences still include long grammatical sequences, we have taken two different measures to attempt to analyze them.

Our first measure involved a small modification to our grammar. We introduced a new non-terminal, *JUNK*. *JUNK* was permitted to occur wherever a sentence modifier, a post-nominal modifier, or a pre-nominal adjective could occur. With a fixed penalty, *JUNK* would match a single word or any sequence of words delimited by dashes, parentheses, or brackets. With an additional penalty per word, it could match any sequence of two or more words. This, roughly speaking, should find the grammatical analysis skipping the minimum number of words.

Adding *JUNK* has a slight effect on the set of sentences which parsed in the base run.[2] However, it substantially increased the number of sentences parsed and thus the

[1]There are a few exceptions to this merging; in particular, we do not merge different complement structures because there are subcategorization constraints which check for a specific complement.

[2]Only because the mix of penalties caused one sentence which had previously parsed to hit the edge limit.

parse strategy	# parsed	avg C	avg recall	avg precision
1. base run	227	2.07	45.73	65.89
2. close	228	1.71	49.90	70.85
3. stat	221	1.47	47.77	73.41
4. stat+close	221	1.30	49.51	75.49
5. stat+close+merge	238	1.79	56.08	73.28
6. stat+close+junk	252	1.53	53.63	74.30
7. stat+close+junk+merge	292	2.27	65.45	71.55
8. stat+close+junk+merge+fit	315	3.18	70.60	67.70

Table 1: *Parsing evaluation over entire corpus, showing number of sentences obtaining one or more parses, average number of crossings per sentence, and average recall and precision.*

parse strategy	avg C	avg recall	avg precision
1. base run	2.43	70.28	67.08
1m. base run – best parses	0.99	85.31	81.22
2. close	1.93	76.51	72.27
3. stat	1.86	75.96	73.59
4. stat+close	1.63	78.89	75.72
5. stat+close+merge	1.54	76.65	76.34
6. stat+close+junk	1.63	78.32	75.76
7. stat+close+junk+merge	1.62	78.89	75.98
8. stat+close+junk+merge+fit	1.62	78.89	75.98

Table 2: *Parsing evaluation over 206 sentences, showing average number of crossings per sentence, and average recall and precision.*

parse strategy	system recall	system precision
1. base run	40	59
2. close	43	61
3. stat	43	61
4. stat+close	42	61
5. stat+close+merge	45	62
6. stat+close+junk	44	61
7. stat+close+junk+merge	46	60
8. stat+close+junk+merge+fit	47	60

Table 3: *System performance in template filling task.*

overall recall (Table 1, line 6 vs. line 4). Combining *JUNK* and merging produced even more dramatic improvements (line 7).

6.2.5 Fitted parse

The second measure we introduced to process sentences which still did not parse was the *fitted parse*. The basic idea of a fitted parse is to "cover" the sentence using constituents of a few types [Jensen *et al.*, 1983]. In our implementation, we looked first for the longest initial S and then repeatedly for the longest S or NP in the remaining words. This managed to get some sort of "parse" for all but two sentences in the corpus. Including these in the scoring produced a further jump in total average recall, to over 70%, but — not surprisingly — with a loss of precision and an increase in crossings.

6.2.6 Overview

While all the heuristics do appreciably better than the base run, we can see that, except for the base run, the recall for the subset of 206 sentences (Table 2) varies little (less than 3%). Most of the gain in recall, measured over the entire corpus (Table 1), therefore reflects the ability to analyze a larger number of sentences (without a substantial degradation in parsing quality).

6.3 System performance

Table 3 shows the performance of the entire extraction system for the same 8 runs. For each run, the total recall and precision (in terms of slots filled in the templates) is tabulated. The precision is fairly constant; the system recall correlates very roughly with the average parse tree recall measured over the entire corpus. As we have just noted, the average parse recall is in turn (except for the base run) closely correlated to the total number of sentences parsed.

7 Conclusion

The availability of a substantial file of parses conforming to a relatively well-specified standard affords a number of opportunities for interesting computational linguistic research. One such opportunity is the evaluation of both grammars and parsing strategies, such as we have shown in this paper. We believe that such evaluation will be crucial to the further development of efficient, broad-coverage syntactic analyzers. By restructuring the generated parse trees to conform to those of the standard, and using agreed-upon comparison metrics, it should be possible to make meaningful comparisons across systems, even those using very different strategies.

We intend to apply our evaluation methods to additional parsing heuristics in the near future. In particular, we noted earlier that selectional constraints are implemented as penalties in our system (preference semantics), and these evaluation methods can assist us in obtaining a proper balance of syntactic and semantic penalties.

By extending our approach, it should be possible to produce more detailed diagnostic information about the parser output. For example, by stripping away some of the structure from both the standard and the system-generated parses, it would be possible to focus on the system's performance on NP bracketing or on S bracketing. Such an approach should also allow for meaningful comparisons with some systems (e.g., some fast deterministic parsers) which generate only more local structures and postpone attachment decisions.

References

[Black *et al.*, 1991] Ezra Black, Steven Abney, Dan Flickenger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Fred Jelinek, Judith Klavans, Mark Liberman, Mitch Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. A procedure for quantitatively comparing the syntactic coverage of English. In *Proceedings of the Speech and Natural Language Workshop*, pages 306–311, Pacific Grove, CA, February 1991. Morgan Kaufmann.

[Chitrao and Grishman, 1990] Mahesh Chitrao and Ralph Grishman. Statistical parsing of messages. In *Proceedings of the Speech and Natural Language Workshop*, pages 263–266, Hidden Valley, PA, June 1990. Morgan Kaufmann.

[Chitrao, 1990] Mahesh Chitrao. *Statistical Techniques for Parsing Messages.* PhD thesis, New York University, 1990. Published as Proteus Project Memorandum No. 38, Computer Science Department, New York University.

[Fujisaki, 1984] Tetsunosuke Fujisaki. A stochastic approach to sentence parsing. In *Proc. 10th Int'l Conf. Computational Linguisitics and 22nd Annl. Meeting Assn. Computational Linguistics*, 1984.

[Grishman and Sterling, 1990] Ralph Grishman and John Sterling. Information extraction and semantic constraints. In *Proc. 13th Int'l Conf. Computational Linguistics (COLING 90)*, 1990.

[Harrison *et al.*, 1991] Philip Harrison, Steven Abney, Ezra Black, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Donald Hindle, Robert Ingria, Mitch Marcus, Beatrice Santorini, and Tomek Strzalkowski. Evaluating syntax performance of parser/grammars. In *Proceedings of the Natural Language Processing Systems Evaluation Workshop*, Berkeley, CA, June 1991. To be published as a Rome Laboratory Technical Report.

[Jensen *et al.*, 1983] K. Jensen, G. E. Heidorn, L. A. Miller, and Y. Ravin. Parse fitting and prose fixing: getting a hold on ill-formedness. *Am. J. Computational Linguistics*, 9(3-4):147–160, 1983.

[Santorini, 1991] Beatrice Santorini. Bracketing guidelines for the penn treebank project. Department of Computer Science, University of Pennsylvania, May 1991.

[Sundheim, 1991] Beth Sundheim. Third message understanding evaluation and conference (MUC-3): Phase 1 status report. In *Proceedings of the Speech and Natural Language Workshop*, pages 301–305, Pacific Grove, CA, February 1991. Morgan Kaufmann.