USING BRACKETED PARSES TO EVALUATE A GRAMMAR CHECKING APPLICATION

Richard H. Wojcik, Philip Harrison, John Bremer

Boeing Computer Services Research and Technology Division
P.O. Box 24346, MS 7L–43
Seattle, WA 98124–2964
Internet: rwojcik@boeing.com, pharrison@boeing.com, jbremer@boeing.com

Abstract

We describe a method for evaluating a grammar checking application with hand–bracketed parses. A randomly–selected set of sentences was submitted to a grammar checker in both bracketed and unbracketed formats. A comparison of the resulting error reports illuminates the relationship between the underlying performance of the parser–grammar system and the error critiques presented to the user.

INTRODUCTION

The recent development of broad–coverage natural language processing systems has stimulated work on the evaluation of the syntactic component of such systems, for purposes of basic evaluation and improvement of system performance. Methods utilizing hand–bracketed corpora (such as the University of Pennsylvania Treebank) as a basis for evaluation metrics have been discussed in Black et al. (1991), Harrison et al. (1991), and Black et al. (1992). Three metrics discussed in those works were the *Crossing Parenthesis Score* (a count of the number of phrases in the machine produced parse which cross with one or more phrases in the hand parse), *Recall* (the percentage of phrases in the hand parse that are also in the machine parse), and *Precision* (the percentage of phrases in the machine parse that are in the hand parse).

We have developed a methodology for using hand–bracketed parses to examine both the internal and external performance of a grammar checker. The internal performance refers to the behavior of the underlying system—i.e. the tokenizer, parser, lexicon, and grammar. The external performance refers to the error critiques generated by the system.[1] Our evaluation methodology relies on three separate error reports generated from a corpus of randomly selected sentences: 1) a report based on unbracketed sentences, 2) a report based on optimally bracketed sentences with our current system, and 3) a report based on the optimal bracketings with the system modified to insure the same coverage as the unbracketed corpus. The bracketed report from the unmodified system tells us something about the coverage of our underlying system in its current state. The bracketed report from the modified system tells us something about the external accuracy of the error reports presented to the user.

Our underlying system uses a bottom–up, full–ambiguity parser. Our error detection method relies on including grammar rules for parsing errorful sentences, with error critiques being generated from the occurrence of an error rule in the parse. Error critiques are based on just one of all the possible parse trees that the system can find for a given sentence. Our major concern about the underlying system is whether the system has a correct parse for the sentence in question. We are also concerned about the accuracy of the selected parse, but our current methodology does not directly address that issue, because correct error reports do not depend on having precisely the correct parse. Consequently, our evaluation of the underlying grammatical coverage is based on a simple metric, namely the parser success rate for satisfying sentence bracketings (i.e. correct parses). Either the parser can produce the optimal parse or it can't.

We have a more complex approach to evaluating the performance of the system's ability to detect errors. Here, we need to look at both the

1. We use the term *critique* to represent an instance of an error detected. Each sentence may have zero or more critiques reported for it.

overgeneration and undergeneration of individual error critiques. What is the rate of *spurious critiques*, or critiques incorrectly reported, and what is the rate of *missed critiques*, or critiques not reported. Therefore we define two additional metrics, which illuminate the spurious and missed critique rates, respectively:

Precision: the percentage of correct critiques from the unbracketed corpus.

Recall: the percentage of critiques generated from an ideal bracketed corpus that are also present among those in the unbracketed corpus.

Precision tells us what percentage of reported critiques are reliable, and Recall tells us what percentage of correct critiques have been reported (modulo the coverage).

OVERVIEW OF THE APPLICATION

The Boeing Simplified English Checker (a.k.a. the BSEC, cf. Hoard, Wojcik, and Holzhauser 1992) is a type of grammar and style checker, but it is more accurately described as a 'controlled English checker' (cf. Adriaens 1992). That is, it reports to users on where a text fails to comply with the aerospace standard for maintenance documentation known as Simplified English (AECMA 1989). If the system cannot produce a parse, it prints the message "Can't do SE check." At present, the Checker achieves parses for about 90 percent of the input strings submitted to it.[2] The accuracy of the error critiques over that 90 percent varies, but our subjective experience suggests that most sentence reports contain critiques that are useful in that they flag some bona fide failure to comply with Simplified English.

The NLP methodology underlying the BSEC does not rely on the type of pattern matching techniques used to flag errors in more conventional checkers. It cannot afford simply to ignore sentences that are too complex to handle. As a controlled sublanguage, Simplified English requires that every word conform to specified usage. That is, each word must be marked as 'allowed' in the lexicon, or it will trigger an error critique. Since the standard generally requires that words be used in only one part of speech, the BSEC produces a parse tree on which to judge vocabulary usage as well as other types of grammatical violations.[3] As one would expect, the BSEC often has to choose between quite a few alternative parse trees, sometimes even hundreds or thousands of them. Given its reliance on full–ambiguity parse forests and relatively little semantic analysis, we have been somewhat surprised that it works as well as it does.

We know of few grammar and style checkers that rely on the complexity of grammatical analysis that the BSEC does, but IBM's Critique is certainly one of the best known. In discussing the accuracy of Critique, Richardson and Braden–Harder (1993:86) define it as "the actual 'under the covers' natural language processing involved, and the user's perception." In other words, there are really two levels upon which to gauge accuracy—that of the internal parser and that of the reports generated. They add: "Given the state of the art, we may consider it a blessing that it is possible for the latter to be somewhat better than the former." The BSEC, like Critique, appears to be smarter than it really is at guessing what the writer had in mind for a sentence structure. Most error critiques are not affected by incorrect phrasal attachment, although grossly incorrect parses lie behind most sentence reports that go sour. What we have not fully understood in the past is the extent to which parsing accuracy affects error critiques. What if we could eliminate all the bad parses? Would that make our system more accurate by reducing incorrect critiques, or would it degrade performance by reducing the overall number of correct critiques reported? We knew that the system was capable of producing good error reports from relatively bad parses, but how many of those error reports even had a reasonably correct parse available to them?

2. The 90 percent figure is based on random samplings taken from maintenance documents submitted to the BSEC over the past two years. This figure has remained relatively consistent for maintenance documentation, although it varies with other text domains.

3. The Simplified English (SE) standard allows some exceptions to the 'single part of speech' rule in its core vocabulary of about a thousand words. The BSEC currently does little to guarantee that writers have used a word in the 'Simplified English' meaning, only that they have selected the correct part of speech.

OVERVIEW OF SIMPLIFIED ENGLISH

The SE standard consists of a set of grammar, style, format, and vocabulary restrictions, not all of which lend themselves to computational analysis. A computer program cannot yet support those aspects of the standard that require deep understanding, e.g. the stricture against using a word in any sense other than the approved one, or the requirement to begin paragraphs with the topic sentence. What a program can do is count the number of words in sentences and compound nouns, detect violations of parts of speech, flag the omission of required words (such as articles) or the presence of banned words (such as auxiliary *have* and *be*, etc.). The overall function of such a program is to present the writer with an independent check on a fair range of Simplified English requirements. For further details on Simplified English and the BSEC, see Hoard et al. (1992) and Wojcik et al. (1990).

Although the BSEC detects a wide variety of Simplified English and general writing violations, only the error categories in Table 1 are relevant to this study: Except for illegal comma usage, which is rather uncommon, the above errors are among the most frequent types of errors detected by the BSEC.

To date, The Boeing Company is the only aerospace manufacturer to produce a program that detects such a wide range of Simplified English violations. In the past, Boeing and other companies have created checkers that report on all words that are potential violations of SE, but such 'word checkers' have no way of avoiding critiques for word usage that is correct. For example, if the word *test* is used legally as a noun, the word–checking program will still flag the word as a potential verb–usage error. The BSEC is the only Simplified English checker in existence that manages to avoid this.[4]

As Richardson and Braden–Harder (p. 88) pointed out: "We have found...that professionals seem much more forgiving of wrong critiques, as long as the time required to disregard them is minimal." In fact, the chief complaint of Boeing technical writers who use the BSEC is when it produces too many nuisance errors. So word–checking programs, while inexpensive and easy to produce, do not address the needs of Simplified English writers.

POS	A known word is used in incorrect part of speech.
NON–SE	An unapproved word is used.
MISSING ARTICLE	Articles must be used wherever possible in SE.
PASSIVE	Passives are usually illegal.
TWO–COMMAND	Commands may not be conjoined when they represent sequential activities. Simultaneous commands may be conjoined.
ING	Progressive participles may not be used in SE.
COMMA ERROR	A violation of comma usage.
WARNING/ CAUTION	Warnings and cautions must appear in a special format. Usually, an error arises when a declarative sentence has been used where an imperative one is required.

Table 1. Error Types Detected By The BSEC

THE PARSER UNDERLYING THE BSEC

The parser underlying the Checker (cf. Harrison 1988) is loosely based on GPSG. The grammar contains over 350 rules, and it has been implemented in Lucid Common Lisp running on Sun workstations.[5] Our approach to error critiquing differs from that used by Critique (Jensen, Heidorn, Miller, and Ravin 1993). Critique uses a two–pass approach that assigns an initial canonical parse in so–called 'Chomsky–normal' form. The second pass produces an altered tree that is an-

4. Oracle's recently released *CoAuthor* product, which is designed to be used with the *Interleaf* word processor, has the potential to produce grammatical analyses of sentences, but it only works as a Simplified English word checker at present.

5. The production version of the BSEC is actually a C program that emulates the lisp development version. The C version accepts the same rules as the lisp version, but there are some minor differences between it and the lisp version. This paper is based solely on the lisp version of the BSEC.

notated for style violations. No–parses cause the system to attempt a 'fitted parse', as a means of producing some information on more serious grammar violations. As mentioned earlier, the BSEC generates parse forests that represent all possible ambiguities vis–a–vis the grammar. There is no 'canonical' parse, nor have we yet implemented a 'fitted parse' strategy to reclaim information available in no–parses.[6] Our problem has been the classic one of selecting the best parse from a number of alternatives. Before the SE Checker was implemented, Boeing's parser had been designed to arrive at a preferred or 'fronted' parse tree by weighting grammatical rules and word entries according to whether we deemed them more or less desirable. This strategy is quite similar to the one described in Heidorn 1993 and other works that he cites. In the maintenance manual domain, we simply observed the behavior of the BSEC over many sentences and adjusted the weights of rules and words as needed.

To get a better idea of how our approach to fronting works, consider the ambiguity in the following two sentences:

(1) The door was closed.
(2) The damage was repaired.

In the Simplified English domain, it is more likely that (2) will be an example of passive usage, thus calling for an error report. To parse (1) as a passive would likely be incorrect in most cases. We therefore assigned the adjective reading of *closed* a low weight in order to prefer an adjectival over a verb reading. Sentence (2) reports a likely event rather than a state, and we therefore weight *repaired* to be preferred as a passive verb. Although this method for selecting fronted parse trees sometimes leads to false error critiques, it works well for most cases in our domain.

BRACKETED INPUT STRINGS

In order to coerce our system into accepting only the desired parse tree, we modified it to accept only parses that satisfied bracketed forms.

6. The BSEC has the capability to report on potential word usage violations in no–parses, but the end–users seem to prefer not to use it. It is often difficult to say whether information will be viewed as help or as clutter in error reports.

For example, the following sentence produces five separate parses because our grammar attaches prepositional phrases to preceding noun phrases and verb phrases in several ways. The structural ambiguity corresponds to five different interpretations, depending on whether the boy uses a telescope, the hill has a telescope on it, the girl on the hill has a telescope, and so on.

(3) The boy saw the girl on the hill with a telescope.

We created a lisp operation called *spe*, for "string, parse, and evaluate," which takes an input string and a template. It returns all possible parse trees that fit the template. Here is an example of an *spe* form for (3):

```
(SPE "The boy saw the girl on the hill with a
      telescope."
 (S (NP the boy)
   (VP (V saw)
        (NP (NP the girl)
            (PP on (NP (NP the hill)
                       (PP with a telescope)))))))
```

The above bracketing restricts the parses to just the parse tree that corresponds to the sense in which the boy saw the girl who is identified as being on the hill that has a telescope. If run through the BSEC, this tree will produce an error message that is identical to the unbracketed report—viz. that *boy, girl, hill,* and *telescope* are NON–SE words. In this case, it does not matter which tree is fronted. As with many sentences checked, the inherent ambiguity in the input string does not affect the error critique.

Recall that some types of ambiguity do affect the error reports—e.g. passive vs. adjectival participial forms. Here is how the *spe* operation was used to disambiguate a sentence from our data:

```
(SPE "Cracks in the impeller blades are not permitted"
 (S (NP Cracks in the impeller blades)
   (VP are not (A permitted))))
```

We judged the word *permitted* to have roughly the same meaning as stative 'permissible' here, and that led us to coerce an adjectival reading in the bracketed input. If the unbracketed input had resulted in the verb reading, then it would have flagged the sentence as an illegal passive. It turned out that the BSEC selected the adjective reading in the unbracketed sentence, and there was no difference between the bracketed and unbracketed error critiques in this instance.

METHODOLOGY

We followed this procedure in gathering and analyzing our data: First, we collected a set of data from nightly BSEC batch runs extending over a three month period from August through October 1991. The data set consisted of approximately 20,000 sentences from 183 documents. Not all of the documents were intended to be in Simplified English when they were originally written. We wrote a shell program to extract a percentage–stratified sample from this data. After extracting a test set, we ended up culling the data for duplicates, tables, and other spurious data that had made it past our initial filter.[7] We ended up with 297 sentences in our data set.

We submitted the 297 sentences to the current system and obtained an error report, which we call the *unbracketed report*. We then created *spe* forms for each sentence. By observing the parse trees with our graphical interface, we verified that the parse tree we wanted was the one produced by the *spe* operation. For 49 sentences, our system could not produce the desired tree. We ran the current system, using the bracketed sentences to produce the *unmodified bracketed report*. Next we examined the 24 sentences which did not have parses satisfying their bracketings but did, nevertheless, have parses in the unbracketed report. We added the lexical information and new grammar rules needed to enable the system to parse these sentences. Running the resulting system produced the *modified bracketed report*. These new parses produced critiques that we used to evaluate the critiques previously produced from the unbracketed corpus. The comparison of the unbracketed report and the modified bracketed report produced the estimates of Precision and Recall for this sample.

7. The BSEC filters out tables and certain other types of input, but the success rate varies with the type of text.

RESULTS

Our 297–sentence corpus had the following characteristics. The length of the sentences ranged between three words and 32 words. The median sentence length was 12 words, and the mean was 13.8 words.[8] Table 2 shows the aggregated outcomes for the three reports.

Checker Outcome	**Unbracketed**	**Unmodified Bracketed**	**Modified Bracketed**
NO PARSE	25	49	25
NO ERROR	123	134	137
ONE OR MORE ERRORS	149	114	135
Totals	297	297	297

Table 2: Overview Of The Results

The table shows the coverage of the system and the impact of the spurious parses. The coverage is reflected in the Unmodified Bracketed column, where 248 parses indicates a coverage of 84 percent for the underlying system in this domain. The table also reveals that there were 24 spurious parses in the unbracketed corpus, corresponding to no valid parse tree in our grammar. The Modified Bracketed column shows the effect on the report generator of forcing the system to have the same coverage as the unbracketed run.

Table 3 shows by type the errors detected in instances where errors were reported. The Spurious Error column indicates the number of errors from the unbracketed sentences which we judged to be bad. The Missed Errors column indicates errors which were missed in the unbracketed report, but which showed up in the modified bracketed

8. Since most of the sentences in our corpus were intended to be in Simplified English, it is not surprising that they tended to be under the 20 word limit imposed by the standard.

report. The modified bracketed report contained only 'actual' Simplified English errors.

Category	Un-bracketed Errors	Spurious Errors	Missed Errors	Actual Errors
POS	120	22	7	105
NON-SE	71	6	5	70
MISSING ARTICLE	38	13	1	26
NOUN CLUSTER	30	7	5	28
PASSIVE	17	7	8	18
TWO-COMMAND	14	3	3	14
ING	5	2	0	3
COMMA ERROR	5	4	0	1
WARNING/ CAUTION	2	0	0	2
Total	302	64	29	267

Table 3: Types Of Errors Detected

For this data, the estimate of Precision (rate of correct error critiques for unbracketed data) is (302–64)/302, or 79 percent. We estimate that this precision rate is accurate to within 5 percent with 95 percent confidence. Our estimate of Recall (rate of correct critiques from the set of possible critiques) is (267–29)/267, or 89 percent. We estimate that this Recall rate is accurate to within 4 percent with 95 percent confidence.

It is instructive to look at a report that contains an incorrectly identified error. The following report resulted from our unbracketed test run:

```
If strut requires six fluid ounces or more to fill, find
leakage source and repair.
  Two commands – possible error:
      find leakage source and repair
  Noun errors:
      fill
          Allowed as: Verb
  Verb errors:
      requires
          Use: be necessary
  Missing articles:
      strut
      leakage source
```

The bracketed run produced a no–parse for this sentence because of an inadequacy in our grammar that blocked *fill* from parsing as a verb. Since it parsed as a noun in the unbracketed run, the system complained that *fill* was allowed as a verb. In our statistics, we counted the *fill* Noun error as an incorrect POS error and the *requires* Verb error as a correct one. This critique contains two POS errors, one TWO–COMMAND error, and two MISSING ARTICLE error. Four of the five error critiques are accurate.

DISCUSSION

We learned several things about our system through this exercise. First, we learned that the act of comparing unbracketed and unmodified bracketed sentences revealed worse performance in the underlying system than we anticipated. We had expected there to be a few more no–parses with unmodified bracketing, but not so many more. Second, the methodology helped us to detect some obscure bugs in the system. For example, the TWO–COMMAND and NOUN CLUSTER errors were not being flagged properly in the unmodified bracketed set because of bugs in the report generator. These bugs had not been noticed because the errors were being flagged properly in some sentences. When a system gets as large and complicated as ours, especially when it generates hundreds or thousands of parse trees for some sentences, it becomes very difficult to detect errors that only show up sporadically and infrequently in the data. Our new methodology provided us with a window on that aspect of system performance.

Perhaps a more interesting observation concerns the relationship between our system and one like Critique, which relies on no–parses to trigger a fitted parse 'damage repair' phase. We believe that the fitted–parse strategy is a good one, although we have not yet felt a strong need to implement it. The reason is that our system generates such rich parse forests that strings which ought to trigger no–parses quite frequently end up triggering 'weird' parses. That is, they trigger parses that are grammatical from a strictly syntactic perspective, but inappropriate for the words in their accustomed meanings. A fitted parse strategy would not work with these cases, because the system has no way of detecting weirdness. Oddly enough, the existence of weird parses often has the same effect in error reports as parse fitting in that they generate error critiques which are useful. The more ambiguity a syntactic system generates, the less likely it is to need a fitted parse strategy to handle unexpected input. The reason for this is that the number of grammatically correct, but 'senseless' parses is large enough to get a parse that would otherwise be ruled out on semantic grounds.

Our plans for the use of this methodology are as follows. First, we intend to change our current system to improve deficiencies and lack of coverage revealed by this exercise. In effect, we plan to use the current test corpus as a training corpus in the next phase. Before deploying the changes, we will collect a new test corpus and repeat our method of evaluation. We are very interested in seeing how this new cycle of development will affect the figures of coverage, Precision, and Recall on the next evaluation.

REFERENCES

Adriaens, G. 1992. From COGRAM to ALCOGRAM: Toward a Controlled English Grammar Checker. *Proceedings of the fifteenth International Conference on Computational Linguistics.* Ch. Boitet, ed. Nantes: COLING. Pp. 595–601.

AECMA. 1989. *A Guide for the Preparation of Aircraft Maintenance Documentation in the Aerospace Maintenance Language. AECMA Simplified English.* AECMA Document: PSC–85–16598, Change 5. Paris.

Black, E., S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. *Proceedings of the Fourth DARPA Speech and Natural Language Workshop.* Pp. 306–311.

Black, E., J. Lafferty, Salim Roukos. 1992. Development and Evaluation of a Broad–Coverage Probabilistic Grammar of English–Language Computer Manuals. *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics.* Pp. 185–192.

Gazdar, G., E. Klein, G. Pullum, and I. Sag. 1985. *Generalized Phrase Structure Grammar.* Cambridge, Mass.: Harvard University Press.

Harrison, P. 1988. *A New Algorithm for Parsing Generalized Phrase Structure Grammars.* Unpublished Ph.D. dissertation. Seattle: University of Washington.

Harrison, P., S. Abney, E. Black, D. Flickinger, C. Gdaniec, R. Grishman, D. Hindle, R. Ingria, M. Marcus, B. Santorini, and T. Strzalkowski. 1991. Evaluating Syntax Performance of Parser/Grammars of English. *Proceedings of Natural Language Processing Systems Evaluation Workshop.* Berkeley, California.

Heidorn, G. 1993. Experience with an Easily Computed Metric for Ranking Alternative Parses. In Jensen, Heidorn, and Richardson 1993. Pp. 29–45.

Hoard, J. E., R. H. Wojcik, and K. Holzhauser. 1992. An Automated Grammar and Style Checker for Writers of Simplified English. In P.O. Holt and N. Williams, eds. 1992. Holt, P. O. 1992. *Computers and Writing: State of the Art.* Boston: Kluwer.

Jensen, K. 1993. PEG: The PLNLP English Grammar. In Jensen, Heidorn, and Richardson 1993. Pp. 29–45.

Jensen, K., G. Heidorn, L. Miller, and Y. Ravin. 1993. Parse Fitting and Prose Fixing. In Jensen, Heidorn, and Richardson 1993. Pp. 53–64.

Jensen, K., G. Heidorn, and S. Richardson, eds. 1993. *Natural Language Processing: The PLNLP Approach.* Boston: Kluwer.

Ravin, Y. 1993. Grammar Errors and Style Weaknesses in a Text–Critiquing System. In Jensen, Heidorn, and Richardson 1993. Pp. 65–76.

Richardson, S. and L. Braden–Harder. 1993. The Experience of Developing a Large–Scale Natural Language Processing System: Critique. In Jensen, Heidorn, and Richardson 1993. Pp. 78–89.

Wojcik, R. H., J. E. Hoard, K. Holzhauser. 1990. The Boeing Simplified English Checker. *Proceedings of the International Conference, Human Machine Interaction and Artificial Intelligence in Aeronautics and Space.* Toulouse: Centre d'Etudes et de Recherches de Toulouse. Pp. 43–57.