The Fitted Parse:
100% Parsing Capability in a Syntactic Grammar of English

Karen Jensen and George E. Heidorn

Computer Sciences Department
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

Abstract

A technique is described for performing *fitted* parsing. After the rules of a more conventional syntactic grammar are unable to produce a parse for an input string, this technique can be used to produce a reasonable approximate parse that can serve as input to the remaining stages of processing. The paper describes how fitted parsing is done in the EPISTLE system and discusses how it can help in dealing with many difficult problems of natural language analysis.

Introduction

The EPISTLE project has as its long-range goal the machine processing of natural language text in an office environment. Ultimately we intend to have software that will be able to parse and understand ordinary prose documents (such as those that an office principal might expect his secretary to cope with), and will be able to generate at least a first draft of a business letter or memo. Our current goal is a system for critiquing written material on points of grammar and style.

Our grammar is written in NLP (Heidorn 1972), an augmented phrase structure language which is implemented in LISP/370. The EPISTLE grammar currently uses syntactic, but not semantic, information. Access to an on-line standard dictionary with about 130,000 entries, including part-of-speech and some other syntactic information (such as transitivity of verbs), makes the system's vocabulary essentially unlimited. We test and improve the grammar by regularly running it on a data base of 2254 sentences from 411 actual business letters. Most of these sentences are rather complicated; the longest contains 63 words, and the average length is 19.2 words.

Since the subset of English which is represented in business documents is very large, we need a very comprehensive grammar and robust parser. In the course of this work we have developed some new techniques to help deal with the refractory nature of natural language syntax. In this paper we discuss one such technique: the *fitted* parse, which guarantees the production of a reasonable parse tree for any string, no matter how unorthodox that string may be. The parse which is produced by *fitting* might not be perfect; but it will always be reasonable and useful, and will allow for later refinement by semantic processing.

There is a certain perception of parsing that leads to the development of techniques like this one: namely, that trying to write a grammar to describe explicitly all and only the sentences of a natural language is about as practical as trying to find the Holy Grail. Not only will the effort expended be Herculean, it will be doomed to failure. Instead we take a heuristic approach and consider that a natural language parser can be divided into three parts:

(a) a set of rules, called the *core grammar*, that precisely define the central, agreed-upon grammatical structures of a language;

(b) peripheral procedures that handle parsing ambiguity: when the core grammar produces more than one parse, these procedures decide which of the multiple parses is to be preferred;

(c) peripheral procedures that handle parsing failure: when the core grammar cannot define an acceptable parse, these procedures assign some reasonable structure to the input.

In EPISTLE, (a) the core grammar consists at present of a set of about 300 syntax rules; (b) ambiguity is resolved by using a metric that ranks alternative parses (Heidorn 1982); and (c) parse failure is handled by the fitting procedure described here.

In using the terms *core grammar* and *periphery* we are consciously echoing recent work in generative grammar, but we are applying the terms in a somewhat different way. Core grammar, in current linguistic theory, suggests the notion of a set of very general rules which define universal properties of human language and effectively set limits on the types of grammars that any particular language may have; periphery phenomena are those constructions which are peculiar to particular languages and which require added rules beyond what the core grammar will provide (Lasnik and Freidin 1981). Our current work is not concerned with the meta-rules of a Universal Grammar. But we have found that a distinction between core and periphery is useful even within a grammar of a particular language — in this case, English.

This paper first reviews parsing in EPISTLE, and then describes the fitting procedure, followed by several examples of its application. Then the benefits of parse fitting and the results of using it in our system are discussed, followed by its relation to other work.

Parsing in EPISTLE

EPISTLE's parser is written in the NLP programming language, which works with augmented phrase structure rules and with attribute-value records, which are manipulated by the rules. When NLP is used to parse natural language text, the records describe constituents, and the rules put these constituents together to form ever larger constituent (or record) structures. Records contain all the computational and linguistic information associated with words, with larger constituents, and with the parse formation. At this time our grammar is sentence-based; we do not, for instance, create record structures to describe paragraphs. Details of the EPISTLE system and of its core grammar may be found in Miller et al., 1981, and Heidorn et al., 1982.

A close examination of parse trees produced by the core grammar will often reveal branch attachments that are not quite right: for example, semantically incongruous prepositional phrase attachments. In line with our pragmatic parsing philosophy, our core grammar is designed to produce unique approximate parses. (Recall that we currently have access only to syntactic and morphological information about constituents.) In the cases where semantic or pragmatic information is needed before a proper attachment can be made, rather than produce a confusion of multiple parses we force the grammar to try to assign a single parse. This is usually done by forcing some attachments to be made to the closest, or rightmost, available constituent. This strategy only rarely impedes the type of grammar-checking and style-checking that we are working on. And we feel that a single parse with a consistent attachment scheme will yield much more easily to later semantic processing than would a large number of different structures.

The rules of the core grammar (CG) produce single approximate parses for the largest percentage of input text. The CG can always be improved and its coverage extended; work on improving the EPISTLE CG is continual. But the coverage of a core grammar will never reach 100%. Natural language is an organic symbol system; it does not submit to cast-iron control. For those strings that cannot be fully parsed by rules of the core grammar we use a heuristic *best fit* procedure that produces a reasonable parse structure.

The Fitting Procedure

The fitting procedure begins after the CG rules have been applied in a bottom-up, parallel fashion, but have failed to produce an S node that covers the string. At this point, as a by-product of bottom-up parsing, records are available for inspection that describe the various segments of the input string from many perspectives, according to the rules that have been applied. The term *fitting* has to do with selecting and fitting these pieces of the analysis together in a reasonable fashion.

The algorithm proceeds in two main stages: first, a *head constituent* is chosen; next, *remaining constituents* are fitted in. In our current implementation, candidates for the head are tested preferentially as follows, from most to least desirable:

(a) VPs with tense and subject;
(b) VPs with tense but no subject;
(c) segments other than VP;
(d) untensed VPs.

If more than one candidate is found in any category, the one preferred is the widest (covering most text). If there is a tie for widest, the leftmost of those is preferred. If there is a tie for leftmost, the one with the best value for the parse metric is chosen. If there is still a tie (a very unlikely case), an arbitrary choice is made. (Note that we consider a VP to be any segment of text that has a verb as its head element.)

The fitting process is complete if the head constituent covers the entire input string (as would be the case if the string contained just a noun phrase, for example, "Salutations and congratulations"). If the head constituent does not cover the entire string, remaining constituents are added on either side, with the following order of preference:

(a) segments other than VP;
(b) untensed VPs;
(c) tensed VPs.

As with the choice of head, the widest candidate is preferred at each step. The fit moves outward from the head, both leftward to the beginning of the string, and rightward to the end, until the entire input string has been fitted into a best approximate parse tree. The overall effect of the fitting process is to select the largest chunk of sentence-like material within a text string and consider it to be central, with left-over chunks of text attached in some reasonable manner.

As a simple example, consider this text string which appeared in one of our EPISTLE data base letters:

"Example: 75 percent of $250.00 is $187.50."

Because this string has a capitalized first word and a period at its end, it is submitted to the core grammar for consideration as a sentence. But it is not a sentence, and so the CG will fail to arrive at a completed parse. However, during processing, the CG will have assigned many structures to its many substrings. Looking for a head constituent among these structures, the fitting procedure will first seek VPs with tense and subject. Several are present: "$250.00 is", "percent of $250.00 is", "$250.00 is $187.50", and so on. The widest and leftmost of these VP constituents is the one which covers the string "75 percent of $250.00 is $187.50", so it will be chosen as head.

The fitting process then looks for additional constituents to the left, favoring ones other than VP. It finds first the colon, and then the word "Example". In this string the only constituent following the head is the final period, which is duly added. The complete fitted parse is shown in Figure 1.

The form of parse tree used here shows the top-down structure of the string from left to right, with the terminal nodes being the last item on each line. At each level of the tree (in a vertical column), the head element of a constituent is marked with an asterisk. The other elements above and below are pre- and post-modifiers. The highest element of the trees shown here is FITTED, rather than the more usual SENT. (It is important to remember that these parse diagrams are only shorthand representations for the NLP record structures, which contain an abundance of information about the string processed.)

The tree of Figure 1, which would be lost if we restricted ourselves to the precise rules of the core grammar, is now available for examination, for grammar and style checking, and ultimately for semantic interpretation. It can take its place in the stream of continuous text and be analyzed for what it is — a sentence fragment, interpretable only by reference to other sentences in context.

```
FITTED|---NP------NOUN*---"Example"
      |---":"
      |---VP*|----NP|-----QUANT---NUM*------"75"
      |      |      |-----NOUN*---"percent"
      |      |      |-----PP|-----PREP------"of"
      |      |              |-----MONEY*----"$250.00"
      |      |----VERB*---"is"
      |      |----NP------MONEY*--"$187.50"
      |---"."
```

Figure 1. An example fitted parse tree.

```
FITTED|---NP*|----NP|-----AJP-----ADJ*----"Good"
      |      |      |-----NOUN*---"luck"
      |      |----CONJ*---"and"
      |      |----NP|-----AJP-----ADJ*----"good"
      |             |-----NOUN*---"selling"
      |---"."
```

Figure 2. Fitted noun phrase (fragment).

```
FITTED|---VP*|----AVP|----ADV*----"Secondly"
      |      |       |----","
      |      |----NP|-----AJP-----ADJ*----"the"
      |      |      |-----NP------NOUN*---"Annual"
      |      |      |-----NP------NOUN*---"Commission"
      |      |      |-----NP------NOUN*---"Statement"
      |      |      |-----NOUN*---"total"
      |      |----VERB----"should"
      |      |----VERB*---"be"
      |      |----NP------MONEY*--"$14,682.61"
      |---","
      |---AVP-----ADV*----"not"
      |---NP------MONEY*--"$14,682.67"
      |---"."
```

Figure 3. Fitted sentence with ellipsis.

Further Examples

The fitted parse approach can help to deal with many difficult natural language problems, including fragments, difficult cases of ellipsis, proliferation of rules to handle single phenomena, phenomena for which no rule seems adequate, and punctuation horrors. Each of these is discussed here with examples.

Fragments. There are many of these in running text; they are frequently NPs, as in Figure 2, and include common greetings, farewells, and sentiments. (N.b., all examples in this paper are taken from the EPISTLE data base.)

Difficult cases of ellipsis. In the sentence of Figure 3, what we really have at a semantic level is a conjunction of two propositions which, if generated directly, would read: "The Annual Commission Statement total should be $14,682.61; the Annual Commission Statement total should *not* be *$14,682.67*." Deletion processes operating on the second proposition are lawful (deletion of identical elements), but massive. It would be unwise to write a core grammar rule that routinely allowed negativized NPs to follow main clauses, because:

(a) the proper analysis of this sentence would be obscured: some pieces — namely, the inferred concepts — are missing from the second part of the surface sentence;

(b) the linguistic generalization would be lost: any two conjoined propositions can undergo deletion of identical (recoverable) elements.

A fitted parse such as Figure 3 allows us to inspect the main clause for syntactic and stylistic deviances, and at the same time makes clear the breaking point between the two propositions and opens the door for a later semantic processing of the elided elements.

Proliferation of rules to handle single phenomena. There are some English constructions which, although they have a fairly simple and unitary form, do not hold anything like a unitary ordering relation within clause boundaries. The vocative is one of these:

(a) *Bill*, I've been asked to clarify the enclosed letter.

```
FITTED|---NP------NOUN*---"Bill"
      |---","
      |---VP*|----NP------PRON*---"I"
      |      |----VERB----"'ve"
      |      |----VERB----"been"
      |      |----VERB*---"asked"
      |      |----INFCL|--INFTO---"to"
      |                |--VERB*---"clarify"
      |                |--NP|-----AJP-----ADJ*----"the"
      |                     |-----AJP-----VERB*---"enclosed"
      |                     |-----NOUN*---"letter"
      |---"."
```

Figure 4. Fitted sentence with initial vocative.

```
FITTED|---NP|-----AJP-----ADJ*----"Good"
      |     |-----NOUN*---"luck"
      |---PP|-----PREP----"to"
      |     |-----NP------PRON*---"you"
      |     |-----CONJ*---"and"
      |     |-----NP------PRON*---"yours"
      |---CONJ----"and"
      |---VP*|----NP------PRON*---"I"
      |      |----VERB*---"wish"
      |      |----NP------PRON*---"you"
      |      |----NP|-----AJP-----ADJ*----"the"
      |      |      |-----ADV-----"VERY"
      |      |      |-----ADJ*----"best"
      |      |----PP|-----PREP----"in"
      |             |-----AJP-----ADJ*----"your"
      |             |-----AJP-----ADJ*----"future"
      |             |-----NOUN*---"efforts"
      |---"."
```

Figure 5. Fitted conjunction of noun phrase with clause.

(b) I've been asked, *Bill*, to clarify the enclosed letter.

(c) I've been asked to clarify the enclosed letter, *Bill*.

In longer sentences there would be even more possible places to insert the vocative, of course.

Rules could be written that would explicitly allow the placement of a proper name, surrounded by commas, at different positions in the sentence — a different rule for each position. But this solution lacks elegance, makes a simple phenomenon seem complicated, and always runs the risk of overlooking yet one more position where some other writer might insert a vocative. The parse fitting procedure provides an alternative that preserves the integrity of the main clause and adds the vocative at a break in the structure, which is where it belongs, as shown in Figure 4. Other similar phenomena, such as parenthetical expressions, can be handled in this same fashion.

Phenomena for which no rule seems adequate. The sentence "Good luck to you and yours and I wish you the very best in your future efforts." is, on the face of it, a conjunction of a noun phrase (or NP plus PP) with a finite verb phrase. Such constructions are not usually considered to be fully grammatical, and a core grammar which contained a rule describing this construction ought probably to be called a faulty grammar. Nevertheless, ordinary English correspondence abounds with strings of this sort, and readers have no difficulty construing them. The fitted parse for this sentence in Figure 5 presents the finite clause as its head and adds the remaining constituents in a reasonable fashion. From this structure later semantic processing could infer that "Good luck to you and yours" really means "I express/send/wish good luck to you and yours" — a special case of formalized, ritualized ellipsis.

Punctuation horrors. In any large sample of natural language text, there will be many irregularities of punctuation which, although perfectly understandable to readers, can completely disable an explicit computational grammar. In business text these difficulties are frequent. Some can be caught and corrected by punctuation checkers and balancers. But others cannot, sometimes because, for all their trickiness, they are not really wrong. Yet few grammarians would care to dignify, by describing it with rules of the core grammar, a text string like:

> "Options: A1-(Transmitter Clocked by Dataset) B3-(without the 605 Recall Unit) C5-(with ABC Ring Indicator) D8-(without Auto Answer) E10-(Auto Ring Selective)."

Our parse fitting procedure handles this example by building a string of NPs separated with punctuation marks, as shown in Figure 6. This solution at least enables us to get a handle on the contents of the string.

```
FITTED|---NP------NOUN*---"Options"
      |---":"
      |---NP------NOUN*---"A1"
      |---"-"
      |---"("
      |---NP|-----NP------NOUN*---"Transmitter"
      |     |-----NOUN*---"Clocked"
      |---PP|-----PREP----"by"
      |     |-----NOUN*---"Dataset"
      |---")"
      |---NP------NOUN*---"B3"
      |---"-"
      |---PP*|----"("
      |      |----PREP----"without"
      |      |----AJP-----ADJ*----"the"
      |      |----QUANT---NUM*----"605"
      |      |----NP------NOUN*---"Recall"
      |      |----NOUN*---"Unit"
      |      |----")"
      |---NP------NOUN*---"C5"
      |---"-"
      |---PP|-----"("
      |     |-----PREP----"with"
      |     |-----NP------NOUN*---"ABC"
      |     |-----NP------NOUN*---"Ring"
      |     |-----NOUN*---"Indicator"
      |     |-----")"
      |---NP------NOUN*---"D8"
      |---"-"
      |---PP|-----"("
      |     |-----PREP----"without"
      |     |-----NP------NOUN*---"Auto"
      |     |-----NOUN*---"Answer"
      |     |-----")"
      |---NP------NOUN*---"E10"
      |---"-"
      |---NP|-----"("
      |     |-----NP------NOUN*---"Auto"
      |     |-----NP------NOUN*---"Ring"
      |     |-----NOUN*---"Selective"
      |     |-----")"
      |---"."
```

Figure 6. Fitted list.

Benefits

There are two main benefits to be gained from using the fitted parse approach. First, it allows for syntactic processing — for our purposes, grammar and style checking — to proceed in the absence of a perfect parse. Second, it provides a promising structure to submit to later semantic processing routines. And parenthetically, a fitted parse diagram is a great aid to rule debugging. The place where the first break occurs between the head constituent and its pre- or post-modifiers usually indicates fairly precisely where the core grammar failed.

It should be emphasized that a fitting procedure cannot be used as a substitute for explicit rules, and that it in no way lessens the importance of the core grammar. There is a tight interaction between the two components. The success of the fitted parse depends on the accuracy and completeness of the core rules; a fit is only as good as its grammar.

Results

In December of 1981, the EPISTLE grammar, which at that time consisted of about 250 grammar rules and did *not* include the fitted parsing technique, was run on the data base of 2254 sentences from business letters of various types. The input corpus was very raw; it had not been edited for spelling or other typing errors, nor had it been manipulated in any way that might have made parsing easier.

At that time the system failed to parse 832, or 36%, of the input sentences. (It gave single parses for 41%, double parses for 11%, and 3 or more parses for 12%.) Then we added the fitting procedure and also worked to improve the core grammar.

Concentrating only on those 832 sentences which in December failed to parse, we ran the grammar again in July, 1982, on a subset of 163 of them. This time the number of core grammar rules was 300. Where originally the CG could parse none of these 163 sentences, this time it yielded parses (mostly single or double) for 109 of them. The remaining 54 were handled by the fitting procedure.

Close analysis of the 54 fitted parses revealed that 14 of these sentences bypass the core grammar simply because of missing dictionary information: for example, the CG contains a rule to parse ditransitive VPs (indirect object-taking VPs with verbs like "give" or "send"), but that rule will not apply if the verb is not marked as ditransitive. The EPISTLE dictionary will eventually have all ditransitive verbs marked properly, but right now it does not.

Removing those 14 sentences from consideration, we are left with a residue of 40 strings, or about 25% of the 163 sentences, which we expect *always* to handle by means of the fitted parse. These strings include all of the problem types mentioned above (fragments, ellipsis, etc.), and the fitted parses produced were adequate for our purposes. It is not yet clear how this 25% might extrapolate to business text at large, but it seems safe to say that there will always be a significant percentage of natural business correspondence which we cannot expect to parse with the core grammar, but which responds nicely to peripheral processing techniques like those of the fitted parse. (A more recent run of the entire data base resulted in 27% fitted parses.)

Related Work

Although we know of no approach quite like the one described here, other related work has been done. Most of this work suggests that unparsable or ill-formed input should be handled by *relaxation techniques*, i.e., by relaxing restrictions in the grammar rules in some principled way. This is undoubtedly a useful strategy — one which EPISTLE makes use of, in fact, in its rules for detecting grammatical errors (Heidorn et al. 1982). However, it is questionable whether such a strategy can ultimately succeed in the face of the overwhelming (for all practical purposes, infinite) variety of ill-formedness with which we are faced when we set out to parse truly unrestricted natural language input. If all ill-formedness is *rule-based* (Weischedel and Sondheimer 1981, p. 3), it can only be by some very loose definition of the term *rule*, such as that which might apply to the fitting algorithm described here.

Thus Weischedel and Black, 1980, suggest three techniques for responding intelligently to unparsable inputs:

(a) using presuppositions to determine user assumptions; this course is not available to a syntactic grammar like EPISTLE's;

(b) using relaxation techniques;

(c) supplying the user with information about the point where the parse blocked; this would require an interactive environment, which would not be possible for every type of natural language processing application.

Kwasny and Sondheimer, 1981, are strong proponents of relaxation techniques, which they use to handle both cases of clearly ungrammatical structures, such as *co-occurrence violations* like subject/verb disagreement, and cases of perfectly acceptable but difficult constructions (ellipsis and conjunction).

Weischedel and Sondheimer, 1982, describe an improved ellipsis processor. No longer is ellipsis handled with relaxation techniques, but by predicting *transformations* of previous parsing paths which would allow for the matching of fragments with plausible contexts. This plan would be appropriate as a next step after the fitted parse, but it does not guarantee a parse for all elided inputs.

Hayes and Mouradian, 1981, also use the relaxation method. They achieve flexibility in their parser by relaxing *consistency constraints* (grammatical restrictions, like Kwasny and Sondheimer's co-occurrence violations) and also by relaxing ordering constraints. However, they are working with a restricted-domain semantic system and their approach, as they admit, "does not embody a solution for flexible parsing of natural language in general" (p. 236).

The work of Wilks is heavily semantic and therefore quite different from EPISTLE, but his general philosophy meshes nicely with the philosophy of the fitted parse: "It is proper to prefer the normal...but it would be absurd...not to accept the abnormal if it is described" (Wilks 1975, p. 267). Wilks' approach to machine translation which involves doing some amount of the translation on a phrase-by-phrase basis is relevant here, too. With fitted parsing, it might be possible to get usable translations for strings that cannot be completely parsed with the core grammar by translating each phrase of the fitted parse separately.

Acknowledgements

We would like to thank Lance Miller for his interest and encouragement in this work, and Roy Byrd, Martin Chodorow, Robert Granville and John Sowa for their comments on an earlier version of this paper.

References

Hayes, P.J. and G.V. Mouradian. 1981. "Flexible Parsing" in *Am. J. Comp. Ling.* 7.4, 232-242.

Heidorn, G.E. 1972. "Natural Language Inputs to a Simulation Programming System." Technical Report NPS-55HD72101A. Monterey, Cal: Naval Postgraduate School.

Heidorn, G.E. 1982. "Experience with an Easily Computed Metric for Ranking Alternative Parses" in *Proc. 20th Annual Meeting of the ACL.* Toronto, Canada, 82-84.

Heidorn, G.E., K. Jensen, L.A. Miller, R.J. Byrd, and M.S. Chodorow. 1982. "The EPISTLE Text-Critiquing System" in *IBM Sys. J.* 21.3, 305-326.

Kwasny, S.C. and N.K. Sondheimer. 1981. "Relaxation Techniques for Parsing Ill-Formed Input" in *Am. J. Comp. Ling.* 7.2, 99-108.

Lasnik, H. and R. Freidin. 1981. "Core Grammar, Case Theory, and Markedness" in *Proc. 1979 GLOW Conf.* Pisa, Italy.

Miller, L.A., G.E. Heidorn and K. Jensen. 1981. "Text-Critiquing with the EPISTLE System: An Authors's Aid to Better Syntax" in *AFIPS Conf. Proc.*, Vol. 50. Arlington, Va., 649-655.

Weischedel, R.M. and J.E. Black. 1980. "Responding Intelligently to Unparsable Inputs" in *Am. J. Comp. Ling.* 6.2, 97-109.

Weischedel, R.M. and N.K. Sondheimer. 1981. "A Framework for Processing Ill-Formed Input." Research Report. Univ. of Delaware.

Weischedel, R.M. and N.K. Sondheimer. 1982. "An Improved Heuristic for Ellipsis Processing" in *Proc. 20th Annual Meeting of the ACL.* Toronto, Canada, 85-88.

Wilks, Yorick. 1975. "An Intelligent Analyzer and Understander of English" in *Comm. ACM* 18.5, 264-274.