Parsing for Grammar and Style Checking

Gregor Thurmair
DI AP 323
Siemens AG
Internet: metal@ztivax.siemens.com
uucp: mcsun!unido!ztivax!metal

1. Abstract

The following paper describes some basic problems which have to be tackled if a morphosyntactic parser is to be configured in a grammar and style checking environment. Whereas grammar checking has to deal with ill-formed input which by definition is outside the scope of a grammar, style checking has problems in grammar coverage and intentionality of style.

To overcome these problems, a method is presented based on the METAL grammar formalism which allows for fallback rules, levelling and scoring mechanisms, and other features which can be used. It will be described what kinds of information and processing are needed to implement such checkers.

Finally, some examples are given which illustrate the mode of operation of the method described.

2. The problem domain

There is a fundamental difference between grammar and style checking: Grammar checking tries to find ill-formedness which by definition is considered to be a mistake and MUST be corrected; style checking has to do with well-formed but somehow marked text. As a result, style checking has to be much more "liberal" as it has to do with "deviations" which might have been intended by the author, but CAN be corrected. This results in two different sets of requirements for a parser.

Concerning a grammar checker, its task is outside the scope of a grammar by definition: A grammar tries to describe (only and exactly) the grammatical structures of a language. Every ungrammatical sentence should cause a parse failure.

Moreover, to detect a grammar error, the parser has to successfully parse a given sentence. In order to parse it, however, information must be used which could have been violated. E.g. in (1) (example from German), agreement is the only way to decide which NP is subject (namely the second) and which is object; (2) is ambiguous as both NPs are plural:

1. Die Tiger tötet der Mann
   (the tigers kills the man)
2. Die Tiger töten die Männer
   (the tigers kill the men)

If agreement is violated it is hard to find out what the subject should be; and therefore it is hard to detect that agreement is violated.

The "circulus vitiosus" is that the parser should detect errors the correct interpretation of which is needed to obtain an overall parse on the basis of which the error can be detected.

There is an additional problem with grammar checking: If the grammar becomes more com-

plex, several competing parses for a given sentence might be found. Diagnosis then depends on what parse has been chosen. The application (checking of larger texts) does not allow for asking the user which interpretation to pick; the parser has to find the "best path" and interpret it. This might lead to the result that sentences are flagged which are correct (from the user's point of view) but did not result in the "best path" parse. E.g. if a PP can be argument of a noun as well as a verb, different flags might be set depending on which reading "wins".

Style checking has a different set of problems to solve. First it has to be found out, what "style" is, i.e. what has to be checked. The present paper will not contribute to this debate; we take as input guidelines which are used in the process of technical writing and in the production of technical documents (cf. Schmitt 89).

These guidelines have to be "translated" into a operational form; e.g. what should be checked if the user is asked not to write "too complex" sentences? In 4. below, some examples of phenomena are given which should be marked.

As style is a kind of producing non-standard structures (i.e. structures which are not covered by standard grammars), we need a powerful parser and a grammar with large coverage to interpret style phenomena; i.e. the linguistic structures which have to be interpreted for style phenomena can and will be very complex. Also, the risk of parse failure will increase, and we need a kind of "post mortem" diagnosis for cases which could not be handled. We need a parser which allows for that.

As far as diagnosis is concerned, the checker should be cautious and formulate questions rather than correct things, as a stylistic variant could be intended by the text author. It also should not mark too many things; e.g. if the rule is "avoid passives" it should certainly not flag every passive sentence. I.e. the diagnostics require practical tuning to be really useful.

3. Properties of a parser for style and grammar checking purposes

3.1 Grammar checking

A parser for grammar checking should have the following features:

It should be able to allow for the analysis of parse failures. Compared to an ATN (cf. Weischedel 1982), where a failure ends with the starting state, a chart keeps all the intermediate results and is well suited for diagnostics.

However, diagnostics follow specific information: The diagnosis must know "what to look for" (e.g. wrong agreement, wrong punctuation etc.). It therefore will cover only a part of the potential grammar errors.

Such a "two step approach" has been implemented in the CRITIQUE system (cf. Ravin 1988), where a parse failure is more closely looked at. However, one could think of special "fallback rules" which implement these diagnostics already in the grammar. This means to enlarge the coverage of the grammar for explicitly ungrammatical structures which during parsing could be marked as ungrammatical. This would be just a different kind of representing the diagnosis knowledge but it would be computationally more effective as it could be integrated into the parse itself, leading to a "one step approach".

In this approach, we do not want the fallback rules to fire except if all other rules failed; i.e. we have to avoid that rules which build grammatical structures are not selected, but rules which are meant as fallback rules fire in "regular" parses. Therefore we must be able to build SETs of rules which can be controlled by the grammar writer. We then can fire the sets which build grammatical structures first, and the fallback rules later on. Then we only need to mark the nodes built by the fallback rules

with a flag indicating that there was a fall-back rule (and of what kind it was).

Moreover, as only the "best first" strategy can be applied in this application area, we must be able to tune the parser in such a way that the most plausible reading comes out first. This can be done by a proper scoring mechanism which should be accessible to the grammar writer. This cannot always avoid that the "intended" parse differs from the "best" one, but it at least makes the parsing process more stable and independent of system-internal determinants (like rule order, parsing strategy etc.)

Finally, we must be able to change the error detected by local operations. These operations consist in changing, adding or deleting feature-value pairs or nodes etc. The alternatives here are: Overwrite the respective piece of information by the correct one and re-generate the whole morphosyntactic surface structure; or exchange just a partial structure. This will depend on the kind of error detected.

3.2 Style checking

Instead of discussing what style might be, we concentrate on "bad style" phenomena mentioned in texts on technical writing (cf. Schmitt 89). Examples of bad style are:

- too long sentences
- too complex sentences
- too many prenominal modifiers
- inconsistent terminology
- unclear prepositional phrase relations etc. (these are, of course, language specific)

These criteria have to be reformulated in formal terms of linguistic descriptions, e.g. complexity of sentences could be:

- number of rules fired to parse it
- number of nodes in a tree
- number of nodes of a certain property (e.g. subclauses) etc.

These formal specifications then have to be used in the diagnosis part.

Here again we have the choice between a "two step" approach which first parses and then does diagnostics, or a "one step" approach which does everything during parsing. We could do diagnosis on partial structures and mark the nodes which have been built. If these nodes are used by the parser to build higher non-terminal nodes, the flags are valid; if the nodes are rejected by the parser they are just ignored.

As using bad style does not lead to ungrammatical sentences, we should not need additional grammar rules for style checking. But what we need is a set of flags which are attached to the nodes in question as soon as some diagnosis succeeds. This could be an additional feature set which is set on top of the features used in the regular grammar. It is used to INTERPRET the rules which have fired according to stylistic criteria.

These features have to be kept local to allow for error localization: If the user is told "too complex word" then the system should be able to localize this word in the tree. On the other hand, we need some global information as well which is related either to a sentence as a whole or even the whole text. (If we want we can even compute overall stylistic scores out of them as soon as we know what that means).

They also should be able to be easily added or removed from the grammar, i.e. should be kept as an independent module which simply is not added if the grammar is used for other purposes. Therefore, we need flexible feature maintenance possibilities.

3.3 The METAL grammar as basic tool

Although originally developed for machine translation, the METAL system can fulfill all the requirements mentioned above:

- it is language independent, i.e. it has a common software kernel which interprets the different language knowledge sources. It also takes care of problems like separation of text and layout information in a given text, treatment of editor specific information, etc.
- it uses an active chart as control structure and does some parse failure diagnosis already (for MT purposes), and it stores those tests which did not succeed and prevented a rule from firing to enable later diagnosis
- it has large grammars and lexica for several languages; therefore considerable coverage is available. Also, some fallback rules already exist. Moreover, the rule structure is such that the analysis parts can easily be separated from the translation parts and enriched by other purpose components (like grammar and style checking) (cf. Thurmair 1990)
- it has a special levelling and preferencing mechanism which allows to group rules into levels and use these levels together with explicit scores for good or bad partial parses to control the overall behavior of the parser according to linguistic needs
- it treats nodes as complex bundles of features and values; and it allows for easy feature manipulation (e.g. percolating, unifying, adding etc.) using a set of grammar operators
- it does not only allow for simple tests (e.g. presence of a feature) but also for complex tests, e.g. on structural descriptions of tree structures
- it has to be modified, however, by adding a component which at the end of a parse collects the grammatical and stylistic flags and evaluates them if necessary

4. Some examples

The following section gives some examples and shows how they could be treated. They are taken from German because the need for full parsing is more obvious here than for English. They try to implement some of the technical writers' requirements.

4.1 Conditional clauses without subordinative conjunction

They can be recognized by searching for a subclause which has the finite verb in the first position:

3. Kommt er, (so) gehen wir
   (Comes he, (so) go we)
4. Lesen Sie die Daten ein, schreibt das Programm eine Fehlermeldungen
   (Read you the data in, writes the program an error message)

Conditional clauses like (3) and (4) share the property of having the verb in first position with infinitives, however. Sometimes it is hard to distinguish between both cases: (5) is conditional, (6) is imperative:

5. Geben Sie "Ja" ein, beenden Sie das Programm
   (Enter you "Yes" in, finish you the program)
6. Geben Sie "Ja" ein, beenden Sie das Programm und schalten Sie das Gerät aus
   (Enter you "Yes" in, finish you the program and switch you the device off)

As the conditionals just mentioned are absolutely grammatical in German, the grammar must have a rule that covers this case (i.e. that a subclause can consist of a clause without subordinate conjunction if the verb is first).

The only thing to do for a style checking device is to mark the subclause node for having been built by a rule which is bad from a stylistic point of view. This could be done by putting an appropriate feature onto this node. If

this node contributes to the overall parse (as in (5)) this feature is evaluated; else (as in (6)) it is not.

4.2 Chains of prepositional phrases

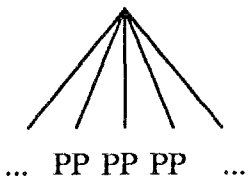
These problems are well known in linguistics. Cases like (7) have unclear references, and not just for the machine! Therefore, chains of PPs should be avoided:

7. The data were input for processing in machine internal format in binary form

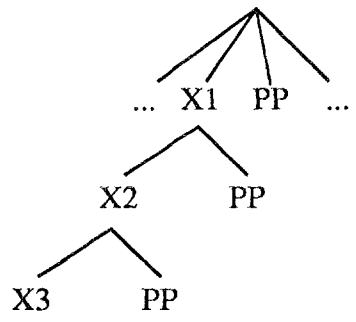
Here the parser will find a solution (e.g. attaching the first two PPs to the NP, the third to clause level), but it will have trouble to do so. "Trouble" might be indicated by many PP-attaching rules being fired; and even if not all of them are successful, some will be, and attachment on different levels is still possible.

In this case, the system cannot simply check the input linear precedence order (as PPs are nonterminal nodes), but we also cannot rely on all PPs being attached as sisters of each other like in (8); cases like (9) are much more likely; and then there is no direct precedence between the PPs any more as precedence holds between X1 and PP, X2 and PP and X3 and PP respectively.

8.



9.



We need some intermediate level here on which a notion like PP is already known, and precedence relations can be determined independent of the actual attachment of these PPs. This requires complex structural matching processes on the trees.

Ambiguity of conjunction falls into the same class of problems: Here again, the parser will finally decide somehow, i.e. try to resolve the ambiguity e.g. of (10).

10. The data were input and output compatible

Again, the question is, how difficult this will be; and this can be expressed in terms of how many rules a conjunctional terminal node can feed (whether successfully or not). In order to know this, we have to examine the chart (as most of the rules tried will not have led to a successful parse) and mark the conjunction accordingly.

4.3 Subject-Object inversion

This last example shows possible complex interactions in the area of style checking. In German, the direct object of a verb can be put before the subject, e.g. in (11), (12), (13):

11. Den Mann hat er gesehen
    (the man has he seen)
12. Die Daten beschreibt das Programm
    (the data describes the program)
13. Die Daten beschreiben die Programme
    (the data describe the programs)

All these sentences are grammatical and have to be covered by the verb valency routines. Sometimes, however, the subject-object-conversion leads to unclear references (as in (13) where both NPs can be both subject and object). This is considered to complicate the process of text understanding. A style checker could flag these occurrences; but there is the following interference:

If the grammar can recognize subject-object-inversion easily (as in (11)), then the reader

can do so as well, and a style checker should not flag anything. The cases which might be ambiguous for the reader, however, are ambiguous for the grammar as well; and as the parser uses certain heuristics to decide on subject and object in unclear cases, it might pick the wrong distribution and not flag anything, although it should do so in exactly this case. The result is that the checker's flagging is useless in the cases where the recognition is good, and that there is no flagging in the real important cases.

This example shows that much fine tuning is necessary to make a checking device a really useful tool and improve its value to users.

5. References

- Chandler, B., 1989: Grammar problems? in: Electric Word 15

- Fink, P., Biermann, A.W., 1986: The Correction of Ill-Formed Input using History-Based Expectation with Applications to Speech Understanding. in: Computational Linguistics 12,1

- Gebruers, R., 1988: Valency and MT, recent developments in the METAL system. in: Proc. 2nd applied ACL, Austin, Tx

- Jensen, K., Binot, J.-L., 1987: Disambiguating Prepositional Phrase Attachments by Using On-Line Dictionary Definitions. in: Computational Linguistics, 13,3/4

- MacDonald, H.H., Frase, L.T., Gingrich, P., Keenan, S.A., 1982: The WRITER'S WORKBENCH: Computer Aids for Text Analysis. in: IEEE Transactions on Communication 30

- Ravin, Y., 1988: Grammar Errors and Style Weaknesses in a text- Critiquing System. in: IEEE Transactions on Communication 31,3

- Schmitt, H., 1989: Writing Understandable Technical Texts. Esprit 2315 Report.

- Thurmair, G., 1990: Recent developments in Machine Translation. (to appear in: Computers and Humanities)

- Weischedel, R.M., Sondheimer, N.K., 1983: Meta-rules as a basis for processing ill-formed input. in: ACL 9

- Weischedel, R.M., Ramshaw, L.A., 1987: Reflections on the knowledge needed to process ill-formed language. In: Nirenburg, S., ed.: Machine Translation, Theoretical and methodological issues. Cambridge Univ. Press, 155-167