Coping with Extragrammaticality

Jaime G. Carbonell and Philip J. Hayes

Computer Science Department, Carnegie-Mellon University
Pittsburgh, PA 15213, USA

Abstract[1]

Practical natural language interfaces must exhibit robust behaviour in the presence of extragrammatical user input. This paper classifies different types of grammatical deviations and related phenomena at the lexical and sentential levels, discussing recovery strategies tailored to specific phenomena in the classification. Such strategies constitute a tool chest of computationally tractable methods for coping with extragrammaticality in restricted domain natural language. Some of the strategies have been tested and proven viable in existing parsers.

1. Introduction

Any robust natural language interface must be capable of processing input utterances that deviate from its grammatical and semantic expectations. Many researchers have made this observation and have taken initial steps towards coverage of certain classes of extragrammatical constructions. Since robust parsers must deal primarily with input that does meet their expectations, the various efforts at coping with extragrammaticality have generally been structured as extensions to existing parsing methods. Probably the most popular approach has been to extend syntactically-oriented parsing techniques employing Augmented Transition Networks (ATNs) [21, 24, 25, 29]. Other researchers have attempted to deal with ungrammatical input through network-based semantic grammar techniques [19, 20], through extensions to pattern matching parsing in which partial pattern matching is allowed [16], through conceptual case frame instantiation [12, 22], and through approaches involving multiple cooperating parsing strategies [7, 9, 18].

Given the background of existing work, this paper focuses on two major objectives:

1. to create a taxonomy of grammatical deviations covering a broad range of extragrammaticalities,
2. to outline strategies for processing many of these deviations,
3. to assess how easily these strategies can be employed in conjunction with existing parsing methods.

The overall result should be a synthesis of different parse-recovery strategies organized by the grammatical phenomena they address (or violate), an evaluation of how well the strategies integrate with existing approaches to parsing extragrammatical input, and a set of characteristics desirable in any parsing process dealing with extragrammatical input. We hope this will aid researchers designing robust natural language interfaces in two ways:

1. by providing a tool chest of computationally effective approaches to cope with extragrammaticality;
2. by assisting in the selection of a basic parsing methodology in which to embed these recovery techniques.

In assessing the degree of compatibility between recovery techniques and various approaches to parsing, we will avoid the issue of whether a given recovery technique ***can*** be used with a specific approach to parsing. The answer to such a question is almost always affirmative. Instead, we will be concerned with how ***naturally*** the recovery strategies fit with the various parsing approaches. In particular, we will consider the computational tractability of the recovery strategies and how easily they can obtain the information they need to operate in the context of different parsing approaches.

Extragrammaticalities include patently ungrammatical constructions, which may nevertheless be semantically comprehensible, as well as lexical difficulties (e.g. misspellings), violations of semantic constraints, utterances that may be grammatically acceptable but are beyond the syntactic coverage of the system, ellipsed fragments and other dialogue phenomena, and any other difficulties that may arise in parsing individual utterances. An *extragrammaticality* is thus defined with respect to the capabilities of a particular system, rather than with respect to an absolute external competence model of the ideal speaker. Extragrammaticality may arise at various levels: lexical, sentential, and dialogue. This paper addresses the first two categories; the third is discussed in [8, 11]. Our discussions are based on direct experience with various working parsers: FLEXP, CASPAR and DYPAR [7, 8, 16].

2. Lexical Level Extragrammaticalities

One of the most frequent parsing problems is finding an unrecognizable word in the input stream. The following sections discuss the underlying reasons for the presence of unrecognizable words and describe suitable recovery strategies.

2.1. The unknown word problem

The word is a legitimate lexeme but is not in the system's dictionary. There are three reasons for this:

- The word is outside the intended coverage of the interface (e.g. There is no reason why a natural language interface to an electronic mail system should know words like "chair" or "sky", which cannot be defined in terms of concepts in its semantic domain).

[1]This research was sponsored in part by the Air Force Office of Scientific Research under Contract AFOSR-82-0219 and in part by Digital Equipment Corporation as part of the XCALIBUR project.

- The word refers to a legitimate domain concept or combination of domain concepts, but was not included in the dictionary. (e.g. A word like "forward" [a message] can be defined as a command verb, its action can be clearly specified, and the objects upon which it operates — an old message and a new recipient — are already well-formed domain concepts.)
- The word is a proper name or a unique identifier, such as a catalogue part name/number, not heretofore encountered by the system, but recognizable by a combination of contextual expectations and morphological or orthographic features (e.g., capitalization).

In the first situation, there is no meaningful recovery strategy other than focused interaction [15] to inform the user of the precise difficulty. In the third, little action is required beyond recognizing the proper name and recording it appropriately for future reference. The second situation is more complicated; three basic recovery strategies are possible:

1. Follow the KLAUS [14] approach where the system temporarily wrests initiative from the user and plays a well designed "twenty questions" game, classifying the unknown term syntactically, and relating it semantically to existing concepts encoded in an inheritance hierarchy. This method has proven successful for verbs, nouns and adjectives, but only when they turn out to be instances of predefined general classes of objects and actions in the domain model.
2. Apply the *project and integrate* method [6] to infer the meaning and syntactic category of the word from context. This method has proven useful for nouns and adjectives whose meaning can be viewed as a recombination of features present elsewhere in the input. Unlike the KLAUS method, it operates in the background, placing no major run-time burden on the user. However, it remains highly experimental and may not prove practical without user confirmation.
3. Interact with the user in a focused manner to provide a paraphrase of the segment of input containing the unknown word. If this paraphrase results in the desired action, it is stored and becomes the meaning of the new word in the immediate context in which it appeared. The LIFER system [20] had a rudimentary capacity for defining synonymous phrases. A more general method would distinguish between true synonymy and functional equivalence in order to classify the new word or phrase in different semantic contexts.

2.2. Misspellings

Misspellings arise when an otherwise recognizable lexeme has letters omitted, substituted, transposed, or spuriously inserted. Misspellings are the most common form of extragrammaticality encountered by natural language interfaces. Usually, a word is misspelt into an unrecognizable character string. But, occasionally a word is misspelt into another word in the dictionary that violates semantic or syntactic expectations. For instance:

Copy the flies from the accounts directory to my directory

Although "flies" may be a legitimate word in the domain of a particular interface (e.g., the files could consist of statistics on med-fly infestation in California), it is obvious to the human reader that there is a misspelling in the sentence above.

There are well-known algorithms for matching a misspelt word against a set of possible corrections [13], and the simplest recovery strategy is to match unknown words against the set of all words in an interface's dictionary. However, this obviously produces incorrect results when a word is misspelt into a word already in the dictionary, and can produce unnecessary ambiguities in other cases.

Superior results are obtained by making the spelling correction sensitive to the parser's syntactic and semantic expectations. In the following example:

Add two fixed haed dual prot disks to the order

"haed" can be corrected to: "had", "head", "hand", "heed", and "hated". Syntactic expectations rule two of these out, and domain semantics rule out two others, leaving "fixed head disk" as the appropriate correction. Computationally, there are two ways to organize this. One can either match parser expectations against all possible corrections in the parser's current vocabulary, and rule out spurious corrections, or one can use the parse expectations to generate a set of possible words that can be recognized at the present point and use this as input to the spelling correction algorithm. The latter, when it can be done, is clearly the preferable choice on efficiency criteria. Generating all possible corrections with a 10,000 word dictionary, only to rule out all but one or two, is a computationally-intensive process, whereas exploiting fully-indexed parser expectations is far more constrained and less likely to generate ambiguity. For the example above, "prot" has 16 possible corrections in a small on-line dictionary. However, domain semantics allow only one word in the same position as "prot", so correction is most effective if the list of possible words is generated first.

2.3. Interaction of morphology and misspelling

Troublesome side-effects of spelling correction can arise with parsers that have an initial morphological analysis phase to reduce words to their root form. For instance, a parser might just store the root form of 'directory' and reduce 'directories' to 'directory' plus a plural marker as part of its initial morphological phase. This process is triggered by failing to recognize the inflected form as a word that is present in the dictionary. It operates by applying standard morphological rules (e.g. `-ies => +y`) to derive a root from the inflected form. It a simple matter to check first for inflected forms and then for misspellings. However, if a word is both inflected and misspelt, the expectation-based spelling corrector must be invoked from within the morphological decomposition routines on potentially misspelt roots or inflexions.

2.4. Incorrect segmentation

Input typed to a natural language interface is segmented into words by spaces and punctuation marks. Both kinds of segmenting markers, especially the second, can be omitted or inserted speciously. Incorrect segmentation at the lexical level results in two or more words being run together, as in "runtogether", or a single word being split up into two or more segments, as in "tog ether" or (inconveniently) "to get her", or combinations of these effects as in "runto geth er". In all these cases, it is possible to deal with such errors by extending the spelling correction mechanism to be able to recognize target words as initial segments of unknown words, and vice-versa.

Compound errors, however, present some difficulties. For instance consider the following example where we have both a missing and a spurious delimiter:

Add two du alport disks to the order

After failing in the standard recovery methods, one letter at a time would be stripped off the beginning of the second unrecognizable word ("alport") and added at the end of the first unrecognizable word ("du"). This process succeeds only if at some step both words are recognizable and enable the parse to continue. Migrating the delimiter (the space) backwards as well as forwards should also be attempted between a pair of unknown words,

stopping if both words become recognizable. Of course, additional compounding of multiple lexical deviations (e.g., misspellings, run-on words and split words in the same segment) requires combinatorially inefficient recovery strategies. Strong parser expectations can reduce the impact of this problem, but at some point tradeoffs must be made between resilience and efficiency in compound error recovery.

3. Sentential Level Extragrammaticalities

We examine ungrammaticalities at the sentential level in five basic categories: missing words, spurious words or phrases, out of order constituents, agreement violations, and semantic constraint violations.

3.1. Missing constituents

It is not uncommon for the user of a natural language interface to omit words from his input. The degree of recovery possible from such ungrammaticalities is, of course, dependent on which words were left out. In practice, words whose contribution to the sentence is redundant are often omitted in an attempt to be cryptic or "computer-like" (as in "Copy new files my directory"). This suggests that techniques that fill in the structural gaps on semantic grounds are more likely to be successful than strategies which do not facilitate the application of domain semantics.

A parsing process postulates a missing word error when its expectations (syntactic or semantic) of what should go at a certain place in the input utterance are violated. To discover that the problem is in fact a missing word, and to find the parse structure corresponding to the user's intention, the parsing process must "step back" and examine the context of the parse as a whole. It needs to ignore temporarily the unfulfilled expectations and their contribution to the overall structure while it tries to fulfil some of its other expectations through parsing other parts of the input and integrating them with already parsed constituents. More specifically, the parser needs to delimit the gap in the input utterance, correlate it with a gap in the parse structure (filling in that gap if it is uniquely determined), and realign the parsing mechanism as though the gap did not exist. Such a realignment can be done top-down by predicting the other constituents from the parse structure already obtained and attempting to find them in the input stream. Alternatively, realignment can be done bottom-up by recognizing as yet unparsed elements of the input, and either fitting them into an existing parse structure, or finding a larger structure to subsume both them and the existing structure. This latter approach is essential when the structuring words are missing or garbled.

3.2. Spurious and unrecognizable constituents

Words in an input utterance that are spurious to a parse can arise from a variety of sources:

- **legitimate phrases that the parser cannot deal with:** It is not uncommon for the user of a restricted domain interface to say things that the interface cannot understand because of either conceptual or grammatical limitations. Sometimes, spurious verbosity or politeness is involved:

  Add if you would be so kind two fixed head and if possible dual ported disks to my order.

  Or the user may offer irrelevant (to the system) explanations or justifications, as observed in preparatory experiments for the GUS system [4], e.g.

  I think I need more storage capacity, so add two fixed head dual ported disks to my order.

  Some common phrases of politeness can be recognized explicitly, but in most cases, the only reasonable response is to ignore the unknown phrases, realign the parse on the recognizable input, and if a semantically and syntactically complete structure results, postulate that the ignored segment was indeed redundant. Isolating certifiable noise phrases in the same way as truly spurious input provides the advantage that they can then be recognized at any point in the input without having to clutter the parser's normal processing with expectations about where they might occur.

- **broken-off and restarted utterances:** These occur when people start to say one thing, change their mind, and say another:

  Add I mean remove a disk from my order

  Utterances in this form are more likely to occur in spoken input, but a similar effect can arise in typed input when a user forgets to hit the erase line or erase character key:

  Add remove a disk from my order
  Add a single ported dual ported disk from my order

  Again the best tactic is to discard the broken-off fragment, but identifying and delineating the superseded fragment requires strategies such as the one discussed below.

- **unknown words filling a known grammatical role:** Sometimes the user will generate an incomprehensible phrase synonymous with a constituent the system is perfectly capable of understanding:

  Add a dual ported rotating mass storage device to my order

  Here the system might not know that "rotating mass storage device" is synonymous with "disk". This phenomenon will result in missing words as well as spurious words. If the system has a unique expectation for what should go in the gap, it should (with appropriate confirmation from the user) record the unknown words as synonymous with what it expected. If the system has a limited set of expectations for what might go in the gap, it could ask the user which one (if any) he meant and again record the synonym for future reference. In cases where there are no strong expectations, the system would ask for a paraphrase of the incomprehensible fragment. If this proved comprehensible, it would then postulate the synonymy relation, ask the user for confirmation, and again store the results for future reference.

As for missing constituents, recovery from spurious interjections generally requires "stepping back" and examining the context of the parse as a whole. In this case however, violations of the parser's expectations should result in skipping over the troublesome segments, and attempting to fulfill the expectations by parsing subsequent segments of the input. If this results in a complete parse, the skipped segment may well be spurious. On the other hand, if a gap in the parse structure remains, it can be correlated with the skipped segments to postulate possible constituents and synonomy relations as illustrated above.

In the case of broken-off utterances, there are some more specific methods that allow the spurious part of the input to be detected:

- If a sequence of two constituents of identical syntactic and semantic type is found where only one is permissible, simply ignore the first constituent. Two main command verbs in sequence (e.g., in the "Add remove ..." example above), instantiate the identical sentential case header role in a case frame parser, enabling the former to be ignored. Similarly, two instantiations of the same prenominal case for the "disk" case frame would be recognized as mutually incompatible and the former again ignored. Other parsing strategies can

be extended to recognize equivalent constituent repetition, but case frame instantiation seems uniquely well suited to it.

- Recognize explicit corrective phrases and if the constituent to the right is of equivalent syntactic and semantic type as the constituent at the left, substitute the right constituent for the left constituent and continue the parse. This strategy recovers from utterances such as "Add I mean remove ...", if "I mean" is recognized as a corrective phrase.
- Select the minimal constituent for all substitutions. For instance the most natural reading of:

  Add a high speed tape drive, that's disk drive, to the order

  is to substitute "disk drive" for "tape drive", and not for the larger phrase "high speed tape drive", which also forms a legitimate constituent of like semantic and syntactic type.

3.3. Out of order constituents and fragmentary input

Sometimes, a user will employ non-standard word order. There are a variety of reasons why users violate expected constituent ordering relations, including unwillingness to change what has already been typed, especially when extensive retyping would be required:

Two fixed head dual ported disk drives add to the order

or a belief that a computer will understand a clipped pseudo-military style more easily than standard usage:

two disk drives fixed head dual ported to my order add

Similar myths about what computers understand best can lead to a very fragmented and cryptic style in which all function words are eliminated:

Add disk drive order

instead of "add a disk drive to my order".

These two phenomena, out of order constituents and fragmentary input, are grouped together because they are similar from the parsing point of view. The parser's problem in each case is to put together a group of recognizable sentence fragments without the normal syntactic glue of function words or position cues to indicate how the fragments should be combined. Since this syntactic information is not present, semantic considerations have to shoulder the burden alone. Hence, parsers which make it easy for semantic information to be brought to bear are at a considerable advantage.

Both bottom-up and top-down recovery strategies are possible for detecting and recovering from missing and spurious constituents. In the bottom-up approach, all the fragments are recognized independently, and purely semantic constraints are used to assemble them into a single framework meaningful in terms of the domain of discourse. When the domain is restricted enough, the semantic constraints can be such that they always produce a unique result. This characteristic was exploited to good effect in the PLANES system [23] in which an input utterance was recognized as a sequence of fragments which were then assembled into a meaningful whole on the basis of semantic considerations alone. A top-down approach to fragment recognition requires that the top-level or organizing concept in the utterance ("add" in the above examples) be located. If it can be, the predictions obtainable from it about what else might appear in the utterance can be used to guide and constrain the recognition of the other fragments.

As a final point, note that in the case of out of order constituents, a parser relying on a strict left-to-right scan will have much greater difficulty than one with more directional freedom. In out of order input, there may be no meaningful set of left-to-right expectations, even allowing for gaps or extra constituents, that will fit the input. For instance, a case frame parser that scans for the head of a case frame, and subsequently attempts to instantiate the individual cases from surrounding input, is far more amenable to this type of recovery than one whose expectations are expressed as word order constraints.

3.4. Syntactic and semantic constraint violations

Input to a natural language system can violate both syntactic and semantic constraints. The most common form of syntactic constraint violation is agreement failure between subject and verb or determiner and head noun:

Do the order include a disk drives?

Semantic constraint violations can occur because the user has conceptual problems:

Add a floating head tape drive to the order

or because he is imprecise in his language, using a related object in place of the object he really means. For instance, if he is trying to decide on the amount of memory to include in an order he might say:

Can you connect a video disk drive to the two megabytes?

When what he really means is "... to the computer with two megabytes of memory?".

These different kinds of constraint violation require quite different kinds of treatment. In general, the syntactic agreement violations can be ignored; cases in which agreement or lack of it distinguishes between two otherwise valid readings of an input are rare. However, one problem that sometimes arises is knowing whether a noun phrase is singular or plural when the determiner or quantifier disagrees with the head noun.

Semantic constraint violations due to a user's conceptual problems are harder to deal with. Once detected, the only solution is to inform the user of his misconception and let him take it from there. The actual detection of the problem, however, can cause some difficulty for a parser relying heavily on semantic constraints to guide its parse. The constraint violation might cause it to assume there was some other problem such as out of order or spurious constituents, and look for (and perhaps even find) some alternative and unintended way of putting all the pieces together. This is one case where syntactic considerations should come to the fore.

Semantic constraint violations based on the mention of a related object instead of the entity actually intended by the user will manifest themselves in the same way as the semantic constraint violations based on misconceptions, but their processing needs to be quite different. The violation can be resolved if the system can look at objects related to the one the user mentioned and find one that satisfies the constraints. In the example above, this means going from the memory size to the machine that has that amount of memory. Clearly, the semantic distance and the type of relationship over which this kind of substitution is allowed needs to be controlled fairly carefully — in a restricted domain everything is eventually related to everything else. Preference rules are needed to control the kind of substitutions that are allowed. In the above example, it might be that a part is allowed to substitute for a whole (metonymy), especially if, as we assumed, the part had been used earlier in the dialogue to distinguish between different instances of the whole.

4. Support for recovery strategies by various parsing approaches

We now turn to the question of incorporating recovery strategies into some of the approaches to parsing found in the literature. We consider three basic classes: transition network approaches (including syntactic ATNs and network-based semantic grammars), pattern matching approaches, and approaches based on case frame instantiation. These classes cover the majority of current parsing systems for restricted domain languages.

All three approaches are able to cope with lexical level problems satisfactorily. However, as we have seen, the application of semantic constraints often makes the correction of lexical problems more efficient and less prone to ambiguity. So parsers that employ semantic constraints (e.g. semantic grammars [20, 5] or case frame instantiation [12, 17]) are more effective in recovery at the lexical level than parsers whose only expectations are syntactic (e.g., purely syntactic ATNs [28]). At the sentential level, however, differences in the abilities of the three approaches to cope naturally with extragrammaticality are far more pronounced. We will examine each approach in turn from this point of view.

4.1. Recovery strategies and transition network parsers

Although attempts have been made to incorporate sentential level recovery strategies into network-based parsers including both syntactically-based ATNs [21, 24, 25, 29] and semantic grammar networks [20], the network paradigm itself is not well suited to the kinds of recovery strategies discussed in the preceding sections. These strategies generally require an interpretive ability to "step back" and take a broad view of the situation when a parser's expectations are violated, and this is very hard to do when using networks. The underlying problem is that a significant amount of state information during the parse is implicitly encoded by the position in the network; in the case of ATNs, other aspects of the state are contained in the settings of scattered registers. As demonstrated by the meta-rule approach to diagnosing parse failures described by Weischedel and Sondheimer [24], these and other difficulties elaborated below do not make recovery from extragrammaticality impossible. However, they do make it difficult and often impractical, since much of the implicitly encoded state must be made declarative and explicit to the recovery strategies.

Often an ATN parse will continue beyond the point where the grammatical deviation, say an omitted word, occurred and reach a node in the network from which it can make no further progress (i.e., no arcs can be traversed). At this point, the parser cannot ascertain the source of the error by examining its internal state even if the state is accessible — the parser may have popped from embedded subnets, or followed a totally spurious sequence of arcs before blocking. If these problems can be overcome and the source of the error determined precisely, a major problem still remains: in order to recover, and parse input that does not accord with the grammar, while remaining true to the network formalism, the parser must modify the network dynamically and temporarily, and use the modified network to proceed through the present difficulties. Needless to say, this is at best a very complex process, one whose computational tractability is open to question in the most general case (though see [21]). It is perhaps not surprising that in one of the most effective recovery mechanisms developed for network-based parsing, the LIFER system's ellipsis handling routine [20], the key step operates completely outside the network formalism.

As we have seen, semantic constraints are very important in recovering from many types of ungrammatical input, and these are by definition unavailable in a purely syntactic ATN parser. However, semantic information can be brought to bear on network based parsing, either through the semantic grammar approach in which joint semantic and syntactic categories are used directly in the ATN, or by allowing the tests on ATN arcs to depend on semantic criteria [2, 3]. In the former technique, the appropriate semantic information for recovery can be applied only if the correct network node can be located — a sometimes difficult task as we have seen. In the latter technique, sometimes known as cascaded ATNs [27], the syntactic and semantic parts of the grammar are kept separate, thus giving the potential for a higher degree of interpretiveness in using the semantic information. However, semantic information represented in this fashion is generally only used to confirm or disconfirm parses arrived at on syntactic grounds and does not participate directly in the parsing process.

A further disadvantage of the network approach for implementing flexible recovery strategies is that networks naturally operate in a top-down left-to-right mode. As we have seen, a bottom-up capability is essential for many recovery strategies, and directional flexibility often enables easier and more efficient operation of the strategies. Of course, the top-down left-to-right mode of operation is a characteristic of the network interpreter, not of the network formalism itself, and an attempt [29] has been made to operate an ATN in an "island" mode, i.e. bottom-up, center-out. This experiment was done in the context of a speech parser where the low-level recognition of many of the input words was uncertain, though the input as a whole was assumed to be grammatical. In that situation, there were clear advantages to starting with islands of relative lexical certainty, and working out from them. Problems, however, arise during leftward expansion from an island when it is necessary to run the network backwards. The admissibility of ATN transitions can depend on tests which access the values of registers which would have been set earlier when traversing the network forwards, but which cannot have been set when traversing backwards. This leads at best to an increase in non-determinism, and at worse to blocking the traversal completely.

4.2. Recovery strategies and pattern matching parsers

A pattern matching approach to parsing provides a better framework to recover from some sentential level deviations than a network-based approach. In particular, the definition of what constitutes a pattern match can be relaxed to allow for missing or spurious constituents. For missing constituents, patterns which match some, but not all, of their components can be counted temporarily as complete matches, and spurious constituents can be ignored so long as they are embedded in a pattern whose other components do match. In these cases, the patterns taken as a whole provide a basis on which to perform the kind of "stepping back" discussed above as being vital for flexible recovery. In addition, when pattern elements are defined semantically instead of lexically, as with Wilks' machine translation system [26], semantic constraints can easily be brought to bear on the recognition. However, dealing with out of order constituents is not so easy for a pattern-based approach since constituent order is built into a pattern in a rigid way, similarly to a network. It is possible to accept any permutation of elements of a pattern as a match, but this provides so much flexibility that many spurious recognitions are likely to be obtained as well as the correct ones (see [16]).

An underlying problem here is that there is no natural way to make the distinctions about the relative importance or difference in role between one word and another. For instance, parsing many of our examples might have involved use of a pattern like:

(<determiner> <disk-drive-attribute> <disk-drive>)*

which specifies a determiner, followed by zero or more attributes of a disk drive, followed by a phrase synonymous with "disk drive". So this pattern would recognize phrases like "a dual ported disk" or "the disk drive". Using the method of dealing with missing constituents mentioned above, "the" would constitute just as good a partial match for this pattern as "disk drive", a clearly undesirable result. The problem is that there is no way to tell the flexible matcher which components of the pattern are discriminating from the point of view of recognition and which are not. Another manifestation of the same problem is that different words and constituents may be easier or harder to recognize (e.g. prepositions are easier to recognize than the noun phrases they introduce), and thus may be more or less worthwhile to look for in an attempt to recover from a grammatical deviation.

The underlying problem is the uniformity of the grammar representation and the method of applying it to the input. Any uniformly represented grammar, whether based on patterns or networks, will have trouble representing and using the kinds of distinctions just outlined, and thus is poorly equipped to deal with many grammatical deviations in an efficient and discriminating manner. See [18] for a fuller discussion of this point.

4.3. Recovery strategies and case frame parsers

Recursive case frame instantiation appears to provide a better framework for recovery from missing words than approaches based on either network traversal or pattern matching. There are several reasons:

- Case frame instantiation is inherently a highly interpretive process. Case frames provide a high-level set of syntactic and semantic expectations that can be applied to the input in a variety of ways. They also provide an overall framework that can be used to realize the notion of "stepping back" to obtain a broad view of a parser's expectations.
- Case frame instantiation is a good vehicle for bringing semantic and pragmatic information to bear in order to help determine the appropriate parse in the absence of expected syntactic constituents. If a preposition is omitted (as commonly happens when dealing with cryptic input from hunt-and-peck typists), the resulting sentence is syntactically anomalous. However, semantic case constraints can be sufficiently strong to attach each noun phrase to the correct structure. Suppose, for instance, the following sentence is typed to an electronic mail system interface:

  Send message John Smith

  The missing determiner presents few problems, but the missing preposition can be more serious. Do we mean to send a message "to John Smith", "about John Smith", "with John Smith", "for John Smith", "from John Smith", "in John Smith", "of John Smith", etc.? The domain semantics of the case frame rule out the latter three possibilities and others like them as nonsensical. However, pragmatic knowledge is required to select "to John Smith" as the preferred reading (possibly subject to user confirmation) — the destination case of the verb is required for the command to be effective, whereas the other cases, if present, are optional. This knowledge of the underlying action must be brought to bear at parse time to disambiguate the cryptic command. In the XCALIBUR system case frame encoding [10], pragmatic knowledge of this kind is represented as preference constraints (cf. [26]) on case fillers. This allows XCALIBUR to overcome problems created by the absence of expected case markers through the application of the appropriate domain knowledge.
- The propagation of semantic knowledge through a case frame (via attached procedures such as those of KRL [1] or SRL [30]), can fill in parser defaults and allow the internal completion of phrases such as "dual disks" to mean "dual ported disks". This process is also responsible for noticing when information is either missing or ambiguously determined, thereby initiating a focused clarificational dialogue [15].
- The representation of case frames is inherently non-uniform. Case fillers, case markers, and case headers are all represented separately, and this distinction can be used by the parser interpretively instantiating the case frame. For instance, if a case frame accounts for the non-spurious part of an input containing spurious constituents, a recovery strategy can skip over the unrecognizable words by scanning for case markers as opposed to case fillers which typically are much harder to find and parse. This ability to exploit non-uniformity goes a long way to overcoming the problems with uniform parsing methods outlined in the previous section on pattern matching.

5. Dialogue Level Extragrammaticality

The underlying causes of many extragrammaticalities detected at the sentential level are rooted in dialogue phenomena. For instance, ellipses and other fragmentary inputs are patently ungrammatical at the sentential level, but can be understood in the context of a dialogue. Viewed at this more global level, ellipsis is not ungrammatical. Nevertheless, the same computational mechanisms required to recover from lexical and (especially) sentential problems are necessary to detect ellipsis and parse the fragments correctly for incorporation into a larger structure. In general, many dialogue phenomena can be classified pragmatically as extragrammaticalities.

In addition to addressing dialogue level extragrammaticalities, any robust parsing system must engage the user in dialogue for cooperative resolution of parsing problems too difficult for automatic recovery. Interaction with the user is also necessary for a cooperative parser to confirm any assumptions it makes in interpreting extragrammatical input and to resolve any ambiguities it cannot overcome on its own. We have referred several times in our discussions to the principle of *focused interaction*, and stated that practical recovery dialogues should be focused as tightly as possible on the specific problem at hand.

Because of space limitations, this paper does not discuss details the automated resolution of dialogue level extragrammaticalities or the use of dialogue to engage the user in cooperative resolution. The interested reader is referred to [8].

6. Concluding Remarks

Any practical natural language interface must be capable of dealing with a wide range of extragrammatical input. This paper has proposed a partial taxonomy of extragrammaticalities that arise in spontaneously generated input to a restricted-domain natural language interface and has presented recovery strategies for handling many of the categories. We also discussed how well three widely employed approaches to parsing — network-based parsing, pattern matching, and case frame instantation — could support the recovery strategies, and concluded that case frame instantiation provided the best basis. The reader is referred to [8]

for a more complete presentation, including a more complete taxonomy and additional recovery strategies, particularly at the dialogue level.

Based on the set of recovery strategies we have examined and the problems that arise in trying to integrate them with techniques for parsing grammatical input, we offer the following set of desiderata for a parsing process that has to deal with extragrammatical input:

- The parsing process should be as interpretive as possible. We have seen several times the need for a parsing process to "stand back" and look at the broad picture of the set of expectations (or grammar) it is applying to the input when an ungrammaticality arises. The more interpretive a parser is, the better able it is to do this. A highly interpretive parser is also better able to apply its expectations to the input in more than one way, which may be crucial if the standard way does not work in the face of an ungrammaticality.
- The parsing process should make it easy to apply semantic information. As we have seen, semantic information is often very important in resolving ungrammaticalities.
- The parsing process should be able to take advantage of non-uniformity in language like that identified in Section 4.2. As we have seen, recovery can be much more efficient and reliable if a parser is able to make use of variations in ease of recognition or discriminating power between different constituents. This kind of "opportunism" can be built into recovery strategies.
- The parsing process should be capable of operating top-down as well as bottom-up. We have seen examples where both of these modes are essential.

We believe that case frame instantiation provides a better basis for parsing extragrammatical input than network-based parsing or pattern matching precisely because it satisfies these desiderata better than the other two approaches. We also believe that it is possible do even better than case frame instantiation by using a multi-strategy approach in which case frame instantiation is just one member (albeit a very important one) of a whole array of parsing and recovery strategies. We argue this claim in detail in [8] and support it by discussion of three experimental parsers that in varying degrees adopt the multi-strategy approach.

7. References

1. Bobrow, D. G. and Winograd, T., "An Overview of KRL, a Knowledge Representation Language," *Cognitive Science*, Vol. 1, No. 1, 1977, pp. 3-46.
2. Bobrow, R. J., "The RUS System," BBN Report 3878, Bolt, Beranek, and Newman, 1978.
3. Bobrow, R. J. and Webber, B., "Knowledge Representation for Syntactic/Semantic Processing," *Proc. National Conference of the American Association for Artificial Intelligence*, Stanford University, August 1980.
4. Bobrow, D. G., Kaplan, R. M., Kay, M., Norman D. A., Thompson, H., and Winograd, T., "GUS: a Frame-Driven Dialogue System," *Artificial Intelligence*, Vol. 8, 1977, pp. 155-173.
5. Brown, J. S. and Burton, R. R., "Multiple Representations of Knowledge for Tutorial Reasoning," in *Representation and Understanding*. Bobrow, D. G. and Collins, A., ed., Academic Press, New York, 1975, pp. 311-349.
6. Carbonell, J. G., "Towards a Self-Extending Parser," *Proceedings of the 17th Meeting of the Association for Computational Linguistics*, 1979, pp. 3-7.
7. Carbonell, J. G. and Hayes, P. J., "Robust Parsing Using Multiple Construction-Specific Strategies," in *Natural Language Parsing Systems*, L. Bolc, ed., Springer-Verlag, 1984.
8. Carbonell, J. G. and Hayes, P. J., "Recovery Strategies for Parsing Extragrammatical Language," *Journal of Computational Linguistics*, Vol. 10, 1984, (publication forthcoming).
9. Carbonell, J. G., Boggs, W. M., Mauldin, M. L. and Anick, P. G., "The XCALIBUR Project, A Natural Language Interface to Expert Systems," *Proceedings of the Eighth International Joint Conference on Artificial intelligence*, 1983.
10. Carbonell, J. G., Boggs, W. M., Mauldin, M. L. and Anick, P. G., "XCALIBUR Progress Report #1: First Steps Towards an Integrated Natural Language Interface," Tech. report, Carnegie-Mellon University, Computer Science Department, 1983.
11. Carbonell, J. G., "Discourse Pragmatics in Task-Oriented Natural Language Interfaces," *Proceedings of the 21st annual meeting of the Association for Computational Linguistics*, 1983.
12. Dejong, G., *Skimming Stories in Real-Time*, PhD dissertation, Computer Science Dept., Yale University, 1979.
13. Durham, I., Lamb, D. D., and Saxe, J. B., "Spelling Correction in User Interfaces," *Comm. ACM*, Vol. 26, 1983.
14. Haas, N. and Hendrix, G. G., "Learning by Being Told: Acquiring Knowledge for Information Management," in *Machine Learning, An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell, eds., Tioga Press, Palo Alto, CA, 1983.
15. Hayes P. J., "A Construction Specific Approach to Focused Interaction in Flexible Parsing," *Proc. of 19th Annual Meeting of the Assoc. for Comput. Ling.*, June 1981, pp. 149-152.
16. Hayes, P. J. and Mouradian, G. V., "Flexible Parsing," *American Journal of Computational Linguistics*, Vol. 7, No. 4, 1981, pp. 232-241.
17. Hayes, P. J. and Carbonell, J. G., "Multi Strategy Construction-Specific Parsing for Flexible Data Base Query and Update," *Proc. Seventh Int. Jt. Conf. on Artificial Intelligence*, Vancouver, August 1981, pp. 432-439.
18. Hayes, P. J. and Carbonell, J. G., "Multi-Strategy Parsing and its Role in Robust Man-Machine Communication," Tech. report CMU-CS-81-118, Carnegie-Mellon University, Computer Science Department, May 1981.
19. Hendrix, G. G., Sacerdoti, E. D. and Slocum, J., "Developing a Natural Language Interface to Complex Data," Tech. report Artificial Intelligence Center., SRI International, 1976.
20. Hendrix, G. G., "Human Engineering for Applied Natural Language Processing," *Proc. Fifth Int. Jt. Conf. on Artificial Intelligence*, 1977, pp. 183-191.
21. Kwasny, S. C. and Sondheimer, N K., "Relaxation Techniques for Parsing Grammatically Ill-Formed Input in Natural Language Understanding Systems," *American Journal of Computational Linguistics*, Vol. 7, No. 2, 1981, pp. 99-108.
22. Schank, R C., Lebowitz, M., Birnbaum, L., "An Integrated Understander," *American Journal of Computational Linguistics*, Vol. 6, No. 1, 1980, pp. 13-30.
23. Waltz, D. L., "An English Language Question Answering System for a Large Relational Data Base," *Comm. ACM*, Vol. 21, No. 7, 1978, pp. 526-539.
24. Weischedel, R. M. and Sondheimer, N. K., "Meta-Rules as a Basis for Processing Ill-formed Input," *Computational Linguistics*, Vol. 10, 1984.
25. Weischedel, R. M. and Black, J., "Responding to Potentially Unparseable Sentences," *American Journal of Computational Linguistics*, Vol. 6, 1980, pp. 97-109.
26. Wilks, Y. A., "Preference Semantics," in *Formal Semantics of Natural Language*, Keenan, ed., Cambridge University Press, 1975.
27. Woods, W. A., "Cascaded ATN Grammars," *American Journal of Computational Linguistics*, Vol. 6, No. 1, August 1980, pp. 1-12.
28. Woods, W. A., Kaplan, R. M., and Nash-Webber, B., "The Lunar Sciences Language System: Final Report," Tech. report 2378, Bolt, Beranek, and Newman, Inc., Cambridge, Mass., 1972.
29. Woods, W. A., Bates, M., Brown, G., Bruce, B., Cook, C., Klovstad, J., Makhoul, J., Nash-Webber, B., Schwartz, R., Wolf, J., and Zue, V., "Speech Understanding Systems - Final Technical Report," Tech. report 3438, Bolt, Beranek, and Newman, Inc., Cambridge, Mass., 1976.
30. Wright, K. and Fox, M., "The SRL Users Manual," Tech. report, Robotics Institute, Carnegie-Mellon University, 1983.