Artificial Intelligence Review **21:** 269–291, 2004.
© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

Parsing Ill-Formed Text Using an Error Grammar

JENNIFER FOSTER & CARL VOGEL
Computational Linguistics Group, Trinity College, University of Dublin, Dublin 2, Ireland (E-mail: jfoster@tcd.ie)

Abstract. This paper presents a robust parsing approach which is designed to address the issue of syntactic errors in text. The approach is based on the concept of an error grammar which is a grammar of ungrammatical sentences. An error grammar is derived from a conventional grammar on the basis of an analysis of a corpus of observed ill-formed sentences. A robust parsing algorithm is presented which is applied after a conventional bottom–up parsing algorithm has failed. This algorithm combines a rule from the error grammar with rules from the normal grammar to arrive at a parse for an ungrammatical sentence. This algorithm is applied to 50 test sentences, with encouraging results.

Keywords: error grammars, Robust parsing, ungrammaticality

1. Introduction

The aim of a robust parser is to behave sensibly when confronted with input which does not conform to its ideas about a particular language. One of the ways in which a parser's expectations can be confounded is if the input contains a syntactic error. The ideal way for a robust parser to behave when confronted with an ungrammatical sentence is to recognize that the sentence is ungrammatical, to suggest possible error diagnoses, and for each diagnosis, to produce appropriate parses.

This paper proposes a robust text parsing approach which is capable of handling a large class of syntactic errors. An overview of the idea is given in Section 2 and comparisons are made to previous attempts to deal with this challenge. A description of the data which forms the basis of this approach is given in Section 3, Section 4 describes the robust parsing algorithm and Section 5 details a preliminary evaluation of the approach and provides suggestions for improvement.

2. Overview

Our approach is to use an empirically grounded model of ungrammaticality or an *error grammar* as the basis of a chart parsing algorithm which is designed to run after an initial bottom–up chart parsing process has failed to deliver a parse for a sentence. This approach contrasts with existing research as discussed below. The error grammar contains a sequence of *error rules*, where each error rule is derived from a conventional rule in such a way that the relationship between them reflects an error that might be expected to occur and the correction that would be applied to make it well-formed. The process of constructing the error grammar involves direct linguistic analysis, constructing information that can be reused in automatic analysis. The accuracy of any machine-learning approach to robust parsing depends on human analysis such as that carried out here. Our recovery chart parsing process uses the edges found during the normal parsing phase together with edges arising from an error rule in its attempt to arrive at a complete parse. In order to keep the search space within a reasonable limit, the interaction between error rules is kept to a minimum. The benefits of this approach to robust parsing are as follows:

(1) *A uniform framework for handling different classes of errors*: When the parsing process consists of the unification of information expressed as constraints on the values of features, *constraint relaxation* is often adopted as a method for introducing robustness. This method proceeds by repeatedly relaxing constraints on feature values until a parse for an ungrammatical sentence can be found (Douglas and Dale, 1992; Vogel and Cooper, 1995; Foster, 2000; Fouvry, 2000, 2003). However, implicit in this approach is the assumption that the number of words in an ungrammatical sentence is the same as the number of words in its grammatical counterpart and, hence, such an approach does not address the problem of errors arising from the omission or insertion of a word within a sentence – two frequent error types according to the analysis carried out as part of this research (see Section 3). Orthogonally, approaches which operate by piecing together the partial parses found during a failed attempt to find a complete parse (Mellish, 1989; Jensen et al., 1983) seem more suited to errors arising from the omission or insertion of words than to errors where the wrong form of a word has been used. In the approach outlined in this paper all errors are dealt with within the uniform framework of an error grammar.

(2) *Limiting the number of robust parses*: Constraint relaxation attempts to solve the problem of ill-formed input quickly become intractable unless strictly controlled. This is because any constraint which fails can potentially be relaxed, leading to nonsensical parses for an ungrammatical sentence. Take, for example, the ungrammatical sentence

`Want to saving money?`

A constraint relaxation approach has the potential to suggest the following as a parse for this sentence:

(S (PRO `want`)(vP (vERB `to`) (nP (dET `saving`) (nOUn `money`))))

One solution to this problem has been proposed by Fouvry (2003) – all the potential parses are generated and then ranked on the basis of a general notion of ‘information loss’. It is not clear how well this works in practice. Douglas and Dale (1992) limit the number of parses by stating in advance what constraints may be relaxed. The approach outlined in this paper is closer in spirit to the latter approach. Error rules are derived on the basis of empirical linguistic data, and are introduced into the parse in a controlled fashion so that nonsensical parses are less likely to occur.

(3) *A clear model of ungrammaticality*: Probabilistic parsers, such as those described in Charniak (2000), are by their very nature robust since they are exposed in their training phase to a large body of empirical data where no explicit distinction is made between the grammatical and the ungrammatical. According to Sampson (2001), this is a good thing since a clear line cannot always be drawn between the two. However, the fact that language errors do undoubtedly occur means that the concept of ungrammaticality cannot be dismissed. The use of an error grammar has the advantage of providing a linguistic model of ungrammaticality, informed by knowledge of constructions and not just likelihood-analysis based on frequencies alone – this means that ill-formed sentences can be diagnosed as such, instead of being viewed merely as sentences occurring with a low frequency. The ability to detect that an error has occurred can sometimes mean the difference between finding a parse for a sentence and finding the *right* parse. Charniak’s parser,[1] for example, returns the following parse for the ill-formed sentence `The closure in computed breadth-first`

(S1 (NP (NP (DT `The`)(NN `closure`)) (PP (IN `in`)
(NP (JJ `computed`)(NN `breadth-first`)))))

The parser fails to recognize that the word `in` is not the preposition but rather a failed attempt to produce the verb `is`. Hence, it constructs a parse which, rather than reflecting the sentence’s meaning, misses the point.

The use of an error grammar is, however, compatible with a probabilistic view of language processing and we do not in any way mean to suggest that our work should be viewed as an alternative to a broad-coverage probabilistic parsing system but rather could be used to provide such a system with an explicit model of ungrammaticality. The error rules we describe in this paper are derived from non-probabilistic grammar rules but there is no reason why they could not be derived from probabilistic grammar rules. The probability of an error rule could then be viewed as a function of the probability of the error it represents (derived from a corpus of ill-formed sentences such as a larger version of the one described in Section 3), and the probability of the normal rule from which it is derived, thus allowing one robust parse to be preferred over another. We take up this discussion again in Section 5.2.

The concept of an error rule is not new. Weischedel and Sondheimer (1983), for example, describe an ATN parser which contains *meta-rules* corresponding to patterns of ill-formedness. Each meta-rule corresponds to a conventional rule and it is invoked during a parse when a conventional rule fails. This approach is very similar in spirit to ours but there are differences. The meta-rules are integrated into the main parsing process, whereas in our approach, the error rules are only applied after a normal parse has failed. The integration of the meta-rules into the main parsing phase is a consequence of the fact that the parser is a top–down ATN rather than a chart parser. An ATN is simultaneously both a parser and a grammar. This means that there is less control over when the meta-rules are invoked, since they can be applied even before sections of the input have been encountered. Another consequence of the ATN-based approach is that the meta-rules are more procedural than declarative – it is difficult to view them in isolation from the actual parsing algorithm and so they cannot be used as a model of ungrammaticality in the same way that the error rules described in this paper can.

Mal-rules or *error productions* are used within the field of applied linguistics to describe the errors typically made by the learners of a language. Schneider and McCoy (1998) describe a chart parsing system (the ICICLE system) which identifies syntactic errors in the writing of native speakers of American Sign Language who are learning English as a second language. This system uses mal-rules to model the errors which are expected to be made by this community. Schneider and McCoy's approach differs from ours in two fundamental ways:

(1) Their attention is restricted to second language errors and they explicitly attempt to model the second language learning process. The class of errors handled by the approach outlined in this paper is more general – this class includes any type of syntactic error, be it language

learning errors or performance errors. James (1998) distinguishes between errors which occur as a result of lack of knowledge of the language, i.e. language learning errors, and those which occur as an oversight. The former are known as *errors* and the latter *mistakes*. In this paper the word 'error' is used with its more general meaning.

(2) The parsing process employed by Schneider and McCoy (1998) is not a two-stage one. This means that when a sentence is being parsed, the set of mal-rules are available along with the set of normal rules from the outset, with the unfortunate consequence that grammatical sentences can be parsed with mal-rules and hence flagged as ungrammatical. Since there are no restrictions on when the mal-rules can be applied and on how many can be applied at any one time, the problem of spiralling spurious ambiguity is also an issue here.

3. Error Data

This section describes how a corpus of syntactic errors was compiled and then used indirectly[2] to generate an error grammar from a conventional grammar.

3.1. *Error Corpus Collection*

In order to implement an error grammar approach to the problem of parsing syntactically ill-formed language, it is necessary to gather information on the kind of errors that a parser might be expected to encounter. A complementary project to this robust parsing study is the compilation of a corpus of erroneous language. The uses of such a corpus are many: as well as telling us what kind of syntactic errors typically occur in written language, it also provides us with a set of authentic sentences which can be used to test the error-handling capability of any robust parsing technique. Outside the realm of language processing, the corpus is a useful source of evidence for any linguist with an interest in linguistic performance as opposed to competence. The corpus currently contains just over 16,000 word tokens.

Inherent in the error collection methodology used to compile this corpus is the assumption that for every ill-formed sentence there is a well-formed one which would have been produced had the source of error been removed.[3] Every time a sentence containing a syntactic error[4] is noted, the following steps are carried out:

1. The sentence is added to the corpus.
2. A note is made of where the sentence occurred.
3. The error in the sentence is diagnosed and based on this diagnosis, the sentence is corrected.
4. The corrected sentence is added to a parallel corpus of well-formed sentences.
5. A note is made of what was done to correct the sentence. For example, to correct the sentence.

> `Are people really capable to understanding them?`

the infinitival marker `to` is replaced by the preposition `of`.

In order to correct an ungrammatical sentence with confidence it is necessary to know what the person responsible for the sentence was trying to say, i.e. it is necessary to know what the ungrammatical sentence means. Every sentence in the corpus is initially encountered as part of the normal reading process which means that the context in which the error occurs is always available. This, in turn, means that most sentences can be corrected without significant risk of ambiguity.[5] If an ungrammatical sentence is encountered whose meaning is not clear, a note is made of the sentence but it is *not* added to the corpus. An example is the sentence

> There is a huge curtailment in the performing arts which has **affected** companies **from** doing productions all over Ireland because they are told to keep costs down.

Even given the context in which the above sentence occurs there still remains some ambiguity: is the sentence suggesting that companies currently doing productions in Ireland have been adversely affected by the cuts, or does it have the stronger reading that the companies have actually been prevented from doing productions?

In some cases, the meaning of an ungrammatical sentence is clear yet there is still more than one way to correct it. An example is the ungrammatical sentence

> Such databases can **be** to some extent **be** improved by reference to non-corpus sources such as the native speaker's knowledge of the language.

This sentence can be corrected by deleting the first occurrence of `be` yielding the grammatical

> Such databases can to some extent be improved by reference to non-corpus sources such as the native speaker's knowledge of the language.

or by deleting the second occurrence yielding the equally grammatical and synonymous

```
Such databases can be to some extent improved by reference
to non-corpus sources such as the native speaker's knowledge
of the language.
```

When there is more than one correction for an ill-formed sentence with each correction expressing the same meaning, all the possible corrections are added to the parallel corpus. This situation holds for approximately 20% of the errors in the corpus.

There are two reasons why the second corpus of corrected sentences is collected: firstly, it serves as an explicit characterization of the systematic ways in which sentences can deviate from what is grammatical, and, secondly, it provides us with a ready-made gold standard which can be used when evaluating any robust parser. According to this evaluation method, an ill-formed sentence is parsed correctly if the parser has found an analysis for it which matches an analysis found for its corrected version (or one of its corrected versions if there are more than one). This evaluation procedure is carried out for the robust parsing strategy described here (see Section 5).

The errors in the corpus come from the following sources, the figure in brackets indicating the number of errors coming from each source: newspapers and magazines (209), emails (169), academic papers and theses (167), websites (67), authors' own drafts (59), student assignments (41), manuals and technical documentation (35), books (28), lecture notes and handouts (15), letters (8), music album sleevenotes (3), public signs (2), teletext (2) and text messages (2). This is quite a broad source of material, especially when compared to other attempts to create a repository of errors: Gojenola and Oronoz (2000) only collect errors in date expressions in student assignments and newspapers; Becker et al. (1999) use an online discussion forum as their only error source.[6]

A difficulty with this approach to the collection of ungrammatical language is that it is time-consuming. The 807 errors which have been collected to date were added to the corpus over a period of 15 months. An alternative approach to this manual one is to use an existing parser to find instances of ungrammaticality. This automatic error collection method would proceed as follows:

1. Find a large body of text. For enough ungrammatical instances to be found, a text with a high frequency of error will be needed.

2. Find a parser with a broad-coverage grammar and lexicon *and* which is capable of distinguishing between grammatical and ungrammatical. The second stipluation rules out most parsers in the probabilistic paradigm.[7]
3. Input the text to the parser. All sentences for which no parse is found are deemed ungrammatical.

An attempt was made to carry out this automatic procedure. This attempt was abandoned, mainly because step two of the procedure proved infeasible in practice. No rule-based parser and broad-coverage grammar could be found which could be used 'off-the-shelf' to parse a large body of raw text. The effort required to get the parser to recognize the well-formed sentences was deemed too much for this approach to be continued. A second problem with this approach is associated with step one of the procedure: limiting one's attention to one particular type of text means that only the error types typically associated with this text type are likely to be found. It is more enlightening to make a broad trawl of different types of written language, time-consuming though it may be.

3.2. *Analysing the Errors*

Every time a sentence is added to the corpus, a note is made of the operation that is needed to correct the sentence. It is this information which is analyzed. Table 1 indicates the correction operators which were applied to the sentences in the corpus. The frequency of each correction operator is provided along with two examples from the error corpus and their corrections from the parallel corrected corpus.

A possible correction operator not listed in Table 1 is the *move* operator which moves a word from the sentence into another position within the sentence. However, since such an operation occurs relatively infrequently (1.5%, according to this corpus), the decision was taken to define it in terms of the add and delete operators, and to treat errors which could be corrected in this way as *composite errors* or errors which can be corrected by applying more than one correction operator. Of course, it is also possible to define the replacement operator in terms of the add and delete operators, but since this is the most frequent correction operator, it is seen as a valid operator in its own right. We will return to this issue in Section 3.3.4.

An interesting point to note is that 89% of the corrections made involved the application of just *one* correction operator. In fact, 92.5%

Table 1. Corpus analysis results

Correction Operator	Examples
Replace a word (49%)	the theory **in** empirical→ the theory **is** empirical staff **was** allowed to return→ staff **were** allowed to return
Add a word (25%)	Will be declaring their undying love for each other? → Will **they** be declaring their undying love for each other? we must assume the validity this induction principle→ we must assume the validity **of** this induction principle
Delete a word (15%)	We **we** have them all→ We have them all a joint development which will **the** provide 10 new apartments→ a joint development which will provide 10 new apartments
More than one of above (11%)	This means **to allow structure-sharing**→ This means **structure-sharing is allowed** What does a single **line yellow** mean?→ What does a single **yellow line** mean?

of the sentences in the corpus can be corrected by applying just one operator. It is these kinds of sentences which will be handled by the robust parsing approach described here. Composite errors are not handled, although they are not incompatible with this approach. Before describing how an error grammar is generated, the replace, add and delete correction operators and the errors associated with them are described in more detail.

3.2.1. *Replacing a Word*

Error corrections which involve the replacement of one word in a sentence for another can be divided into the following categories:

1. *Spelling* (41%): A word can be replaced by a word similar to it in spelling. This is determined as follows: a word X is similar in spelling to a word Y if X can be transformed into Y by changing or deleting at most two letters in X, or by adding at most two letters to X, while keeping the first letter in X unchanged, e.g. `nor` with `not`.
2. *Agreement* (23%): If the word is a noun, verb or determiner, it can be replaced by the same word with a different value for an agreement feature, e.g. first person `am` with third person `is`, singular `man` with plural `men`.
3. *Verb form* (11%): If the word is a verb, it can be replaced by the same verb with a different form, e.g. infinitival `tell` with present participle `telling`.
4. *Same root/different syntactic category* (7%): A word can be replaced by a word with the same lexical root but with a different syntactic category, e.g. adjective `syntactic` with adverb `syntactically`.
5. *Prepositions* (6%): If the word is a preposition, it can be replaced by any other preposition, e.g. `for` with `of`.
6. *Case* (6%): All case marking errors found in the corpus involved the incorrect use of the genitive case on pronouns and nouns, e.g. the noun `rivers` instead of the possessive `river's`.
7. *Auxiliary verbs* (2%): Sometimes the wrong auxiliary verb was used, e.g. the verb `have` was used instead of the copula `be`.
8. *Synonymous verbs* (1.5%): If the word is a verb it can be replaced by a synonymous verb which has a different grammatical distribution, e.g. the verb `comprises` can be replaced by `consists` before the preposition `of`.
9. *Infinitival to and prepositions* (1%): The infinitival marker `to` can be replaced by a preposition and vice versa.
10. *Same root/same part of speech/different meaning* (0.5%): This occurred when two words with the same root and part of speech category but with a similar yet different meaning were confused, e.g. the use of the noun `compilation` instead of the noun `compiler`.
11. *Conjunctions* (0.5%): If the word is a conjunction it can be replaced by another conjunction, e.g. `and` can be replaced by `then` in a sentence beginning with `once`.
12. *Other* (0.5%): There is no obvious similarity between the word to be replaced and the word replacing it, e.g. `when` with `with` or `and` with `of`.

In all but the last of these categories the confusion between the two words can be explained in terms of some kind of similarity between them. An example of an error correction which fits into the last category is the replacement of the word `when` with the word `with` in the following sentence:

```
But when one as doggedly uninteresting as this it seems like
an aeon.
```

3.2.2. *Adding a Word*

Error corrections involving the addition of a word were categorized in terms of the syntactic category of the word that was added:

1. *Determiners* (29%): In all but one of these cases, a determiner needed to be added before a singular noun. The determiners which were added were either `the` (15%) or `a/an` (14%).
2. *Verbs* (22%): In 8% of cases it was an auxiliary verb that needed to be added. In just over half of the 14% of cases involving a missing main verb, the main verb in question was some form of the copula `be`.
3. *Prepositions* (21%): A frequently omitted preposition was the possessive `of` (7%). In another 7% of cases, the omitted preposition was required by another lexical item in the sentence, e.g. the preposition `with` after the verb `deal`. The remaining cases (7%) comprised omissions of the following prepositions: `to`, `by`, `with`, `from`, `in` and `for`.
4. *Pronouns* (11%): Pronouns needing to be added to the object position accounted for 4%, subject pronouns 3%, relative pronouns 3%, reflexive pronouns 0.5% and possessive pronouns 0.5%.
5. *Nouns* (8%): In the majority of cases (6.5%), the noun which is added is preceded by a determiner.
6. *Infinitival* to (6%).
7. *Miscellaneous* (3%): There were one or two omissions of each of the following: adjectives, the complementizer `that`, conjunctions, parts of set phrases such as `how` in `how much` and `such` in `such as`.

3.2.3. *Deleting a Word*

Error corrections involving the deletion of a word were categorized as follows:

1. *Repeated words* (33%): The word which was deleted appeared elsewhere in the sentence. In almost half of these cases, the two occurrences were immediately adjacent.
2. *Repeated similar words* (34%): These kind of deletions are similar to the repeated word deletions, the difference being that the word to be

deleted occurs in the vicinity of another word either of the same syntactic category (31%) or of similar spelling (3%), e.g.:

`Why is do they appear?` or `I don't know but it's its crawling up your leg.`

3. *Unnecessary words* (33%): These kind of deletions involve no repetition and are simply cases where an extraneous word appears in the sentence. Prepositions (10%), determiners (12%) and verbs (5%) make up the majority of these.

3.3. *Generating the Error Grammar*

A grammar of well-formed language is needed in order to generate a grammar of ill-formed. The only constraint on the format of the grammar is that it must be parsable using a chart parser. To test this error grammar approach, a context-free phrase structure grammar containing 1113 rules was used. The non-terminal symbols of the grammar are augmented with agreement features where appropriate. For each error correction type, the process of generating error rules for this type is described.

3.3.1. *Replace a Word*

For each rule in the grammar which expands a pre-terminal symbol (i.e. a part-of-speech category), an error rule is generated for all the words which are similar to the word on the right-hand side of this rule, according to the similarity criteria described in Section 3.2.1. For the rule:

```
verb(sing,third) --> [is]
```

the following error rules are generated:

```
verb(sing,third) spellop [in]
verb(sing,third) spellop [it]
verb(sing,third) spellop [its]
verb(sing,third) spellop [if]
verb(sing,third) agreeop [are]
verb(sing,third) agreeop [am]
verb(sing,third) vformop [be]
verb(sing,third) vformop [being]
verb(sing,third) vformop [been]
```

To distinguish error rules from conventional grammar rules, the −− > connective is replaced by a connective which describes the error, e.g. the connective `spellop` refers to errors which result when the correct and incorrect word are similar in spelling, the connective `agreeop` refers to errors where the correct and incorrect word have conflicting values for an agreement feature, and the connective `vformop` refers to errors where the two words (or verbs) have conflicting verb form values. In all other respects these connectives mean the same thing as the conventional rule connective −− >, which is comparable to that of a standard Definite Clause Grammar (Pereira and Shieber, 1987). 889 error rules of this sort were generated, by hand, availing of linguistic knowledge.

3.3.2. *Add a Word*

For each rule in the grammar (excluding rules expanding pre-terminal symbols), an error rule is generated which has the same right-hand side as the original rule except that a pre-terminal symbol on the right-hand side is removed.[8] So, for example, for the rule

```
np(Num,Per) − − > det(Num,Per),nbar(Num,Per)
```

the following rule is generated:

```
np(Num,Per) missingop nbar(Num,Per)
```

For unary rules where the only thing on the right-hand side is a pre-terminal symbol no error rules with an empty right-hand side are generated. Instead, for each rule which has the pre-terminal symbol somewhere on its right-hand side, a corresponding error rule is generated which omits this category. So, for example, given the unary rule

```
nbar(Num,Per) − − > noun(Num,Per)
```

and the rule

```
np (Num,Per) − − > det(Num,Per),nbar (Num,Per)
```

the following error rule will be generated:

```
np(Num,Per) missingop det(Num,Per)
```

An error rule is not generated if its right-hand side is the same as the right-hand side of a conventional rule, e.g. for the rule

```
nbar(Num,Per) − − > adj, n(Num,Per)
```

the error rule

```
nbar(Num,Per) missingop n(Num,Per)
```

is not generated since this already exists in the grammar as a conventional rule. 335 error rules of this type were generated automatically.

3.3.3. *Delete a Word*

For each rule in the grammar (excluding rules expanding pre-terminal symbols), an error rule is generated which has the same right-hand side as the original rule except that a symbol is added after a preterminal symbol. This symbol must be capable of matching with any pre-terminal symbol in the grammar. Given, for example, the rule

```
vp(Num, Per) − − > v_trans(Num, Per), np(_, _).
```

the following error rule is generated:

```
vp(Num, Per) extraop v_trans(Num, Per), preterm, np(_, _).
```

where `preterm` unifies with every pre-terminal category in the grammar. The rule

```
np(Num, Per) − − > det(Num, Per), nbar(Num, Per)
```

will result in the error rule:

```
np(Num, Per) extraop det(Num, Per), preterm, nbar(Num, Per)
```

and the rule

```
nbar(Num, Per) − − > n(Num, Per)
```

will result in the error rule:

```
nbar(Num, Per) extraop n(Num, Per), preterm
```

The error rules will describe any ill-formed sentence where an extraneous word appears after any word in the well-formed sentence. To allow for an extraneous word at the beginning of a sentence, error rules can be generated from rules which expand sentential categories. So, for example, given the classic sentential rule

```
s − − > np(Num, Per), vp(Num, Per)
```

the following error rule will be generated

```
s extraop preterm, np(Num, Per), vp(Num, Per)
```

403 error rules of this kind were generated automatically.

3.3.4. *Error Grammar Coverage*

The `missingop` and `extraop` error rules are general enough to describe any type of error whose correction involves the deletion of a word from a sentence or the addition of a word to a sentence. As such, these error rules do not reflect the detailed analysis carried out on these kinds of corrections operators (described in Sections 3.2.2 and 3.2.3) but are a generalization of them, ignoring frequency distributional factors. The word replacement error rules are, however, a more direct reflection of the analysis carried out on this operator (see Section 3.2.1) since we have an error rule connective for the first 11 of the categories into which this kind of error correction can be divided. The only exception is the last category where the confusion between two words could not be explained in terms of the similarity between them. The decision is taken to treat such corrections as composite error corrections involving an add and delete operation. Thus, we expect our error grammar, once it has been combined with a suitable parsing algorithm (see Section 4), to be able to parse all errors which are correctable by adding or deleting one word and most errors which are correctable by replacing a word.

4. Robust Recovery Algorithm

In this section we describe our robust recovery algorithm. This algorithm takes as input a sentence which has failed to parse using a conventional bottom–up algorithm, along with the chart edges which were created while trying to find a complete parse. It is important that a bottom–up parsing strategy as opposed to a top–down one is used during the normal parsing phase. A bottom–up parser is driven by the words in the input sentence and will be guaranteed to find all partial parses in the ungrammatical sentence. A top–down parser, on the other hand, is driven by the grammar rules and will run out of steam, leaving sections of the input sentence untouched.

The bottom–up chart parsing algorithm used in our system is shown in Table 2. The chart edge notation (cf. Earley, 1970)

$$\mathtt{CM} - - > \mathtt{CM1} \ldots \mathtt{CMN}* \; [\mathtt{y}, \mathtt{z}]$$
$$\mathtt{C} - - > \mathtt{C1} \ldots *\mathtt{CM} \ldots \mathtt{CN} \; [\mathtt{x}, \mathtt{y}]$$

is explained as follows: a category CM consisting of the sequence of categories CM1 ... CMN has been found from positions y to z in the input string. The * symbol is used to separate categories that have already been found by the parser from those which are still to be found. In the CM edge the * appears after the category sequence CM1 ... CMN which means that this edge has satisfied its expectations and is *inactive*. The second edge is *active* because it is still looking for the categories CM...CN. It has already found the categories C1...CM − 1 between positions x and y in the input string. When it has fulfilled its outstanding expectations, a category C will have been found.

The robust recovery parsing algorithm is given in Table 3. The errorop connective corresponds to any connective which appears in an error rule, e.g. missingop or spellop (see Section 3.3). All the chart edges built during the normal parsing phase, active and inactive, are available to the recovery algorithm. The recovery algorithm works by repeatedly adding new edges corresponding to error rules to the chart and reinvoking the bottom–up chart parsing algorithm to see if a complete parse can be found. After an error rule has been tried (whether successfully or unsuccessfully), the slate is wiped clean for the next error rule. This ensures that there is no interaction between error rules.

Error rules which correspond to the replacement correction operator are chosen on the basis of the actual words in the input sentence, in a process similar to the chart initialization process. Error rules corresponding to the addition and deletion correction operators are chosen in a selection process similar to the bottom–up rule. A forward scan of the edges already found in the chart is also carried out during this selection process so that no new active edges are added to the chart which are not guaranteed to be completed. This forward scan procedure can also be added to the bottom–up rule of chart parsing, thereby further reducing redundant computation.

5. Evaluation

The robust recovery algorithm described in Table 3 was applied to 50 test sentences which were randomly extracted from the error corpus. The shortest sentence in the set of test sentences has 4 words, the longest has 35 words and the average sentence length is 20 words. Before the 50 ill–formed sentences were tested, their corrected versions were parsed using the conventional grammar and the bottom–up chart parsing algorithm described in Table 2.

5.1. *Accuracy*

The recovery process was deemed to have worked if one of the corrections it proposed for an ill-formed sentence matched in structure one of the corrected versions of the sentence. This was the case if the error

Table 2. Bottom–up chart parsing algorithm

```
Bottom–up parse
initialize the chart
WHILE there are more edges on agenda
  E = edge taken from top of agenda
  IF E spans the chart THEN
    report parse
  ELSE
    use E to find more edges using fundamental rule
    use E to find more edges using bottom–up rule
  add E to the chart

Chart initialization
FOR each word W [x,y] in the input string
  FOR each rule $C-- >W$ in the grammar
    add an inactive edge $C - - > W * [x, y]$ to agenda

Fundamental rule
IF there is an inactive edge $CM - - > CM1 \ldots CMN * [y, z]$
  AND
  an active edge $C - - > C1 \ldots * CM \ldots . CN[x, y]$ THEN
    add an edge $C - - > C1 \ldots CM * \ldots . CN[x, z]$ to agenda

Bottom–up rule
IF there is an inactive edge $C1 - - > C11 \ldots C1N * [x, y]$
  AND
  a rule in the grammar $C - - > C1 \ldots CN$ THEN
    add an active edge $C - - > *C1 \ldots CN[x, x]$ to agenda
```

rule used was derived from a conventional rule used at the same point in the parse of the corrected sentence. According to this measure, the robust recovery procedure achieved an accuracy rate of 84%. The 8 failed attempts can be explained as follows:

1. *More than one error in the sentence*: Two sentences in the test data contained two separate errors, e.g. the sentence

> `From all of the above considerations, the following roadmap for the has been derived derived for this presentation.`

which contains a missing word error along with an extra word error. The recovery algorithm (see Table 3) ensures that only one error rule is ever considered during any one parse attempt, with the result that this

Table 3. Recovery algorithm

```
Top–level
 REPEAT
   find an appropriate error rule
   add an edge corresponding to error rule to agenda
   reinvoke the bottom-up chart parser (see Table 2 and
   new bottom–up rule below)
   remove all edges from chart which have arisen
   from this error rule
 UNTIL there are no more appropriate error rules

 Find appropriate error rules (replacement)
 FOR all words W [x,y] in the input string
   FOR each error rule $C$ errorop $W$ in the grammar
     add inactive edge $C$ errorop $W * [x,y]$ to agenda

 Find appropriate error rules (add and delete)
 FOR all inactive edges $C1 - - > C11 \ldots C1N * [x,y]$
 found during the normal parsing phase
   FOR each error rule $C$ errorop $C1 \ldots CN$
     IF there is a sequence of inactive edges starting
       at position y for the categories C2...CN THEN
       Add an active edge $C$ errorop $*C1 \ldots .CN[x,x]$ to agenda

 New bottom–up rule
 IF there is an inactive edge $C1$ errorop $C11 \ldots C1N * [x,y]$
 AND
 there is a rule in the grammar $C - - > C1 \ldots CN$
 AND
 there is a sequence of inactive edges starting at y
 for the categories C2...CN THEN
   add an active edge $C$ errorop $*C1 \ldots CN[x,x]$ to agenda
```

algorithm will not handle sentences containing two or more errors. The next step in this research is to modify the algorithm so that it can consider more than one error rule under certain controlled circumstances.

2. *Composite errors*: Three sentences in the test data contained a composite error, e.g. the sentence

```
But not one of them is capable to deal with robustness
as a whole
```

which can be corrected by replacing `to` with `of` and by replacing `deal` with `dealing`. Sentences containing a composite error will not be dealt with under this approach for the same reason that sentences containing more than one error would not: only one error rule can be considered at any one time. To deal with composite errors, the recovery algorithm will need to be modified so that it has the potential to use more than one error rule at a time, or error rules could be constructed which describe certain types of regularly occurring composite errors.

3. *Failure to recognize a sentence as ungrammatical*: Three sentences in the corpus were not recognized as ill-formed by the normal parser, e.g. the sentence

```
Compared to syntactic valid structures, the set of
syntactically incorrect sentences can be considered almost
infinite
```

which can be corrected by replacing the adjective `syntactic` with the adverb `syntactically`. Another example from the corpus is the sentence

```
Spoken language is not only characterized by hesitations,
breaks and restarts but is additionally narked by different
kinds of syntactic and semantic inconsistencies.
```

where the verb `narked` has been produced instead of the contextually more appropriate `marked`. James (1998) has termed these kinds of errors *covert errors* because the sentences in which they occur are superficially well-formed. If the recovery algorithm was applied to these sentences the appropriate robust parse would be suggested, but since they are not ungrammatical according to the test grammar, the recovery process will never be applied. These kinds of cases will remain a problem until the grammar is extended so that it can take into account sophisticated semantic and contextual information (in itself a formidable task). If the conventional grammar is a probabilistic one, it might be possible to calculate a probabilistic threshold which could be used to determine if an error rule should be fired (see footnote 7).

5.2. *Efficiency*

Table 4 compares the ill-formed sentences which could be parsed correctly to their corrected counterparts, in terms of average number of parses and average number of parse cycles. The notion of a parse cycle was proposed by Mellish (1989) as an implementation-independent measure of a chart parser's efficiency. In one parse cycle, an edge is taken from the agenda, added to the chart and used (via the fundamental and bottom–up rules of chart parsing) to propose more edges.

The increase in number of parses for the set of ill-formed sentences is expected, since the robust recovery algorithm is essentially proposing ways of correcting an ill-formed sentence, and each correction will have a certain number of parses associated with it. Consider, for example, the ill-formed sentence:

`Will be declaring their undying love for each other`

The corrected version of the above sentence is the sentence:

`Will they be declaring their undying love for each other`

and this receives 4 parses. The robust parsing algorithm finds these 4 parses, along with another 4 associated with a well-formed sentence with a different meaning, i.e.:

`They will be declaring their undying love for each other`.

The parse cycle number for an ill-formed sentence includes the parse cycle number for the original parse of the sentence (when no parses are found) plus the parse cycle number for the recovery parse. The large figure of 7180 is also not a mystery given the large number of error rules. The robust recovery algorithm was adapted so that it stopped after one correction had been successfully proposed, i.e. after one error rule had succeeded in finding a set of complete parses for the ill-formed sentence. The average parse cycle number was reduced significantly from 7180 to 1686.

The problem with stopping after one correction is that, for 22% of the sentences in the test data, the first correction obtained did not correspond to the correct one. An example is the `Will be declaring their undying love for each other` sentence discussed above. The correction which posits a pronoun at the very start of the sentence is the one which was suggested first, but which is not the correct one according to our corpus. In order to improve the accuracy associated with proposing just one correction for an ill-formed sentence, it will be necessary

Table 4. Some performance results

	Average parse no.	Average cycle no.
Corrected	5	935
Ill-formed	8	7180

to ensure that the edges corresponding to error rules are added to the chart in a way that mirrors error frequency (Magerman and Weir, 1992). We ordered the rules such that replacement corrections were proposed before addition and deletion corrections since these are the most frequent in the corpus (see Table 1). The individual replacement error rules were also ordered in terms of the frequencies given in Section 3.2.1. A more sophisticated ordering of the error rules which takes into account some of the error analysis results discussed in Sections 3.2.2 and 3.2.3 and not included in our error grammar, is a goal for further research.

Since edges corresponding to error rules are added to the agenda based on the words in the input sentence and the inactive edges found during the initial parsing phase, another way to reduce the number of parse cycles is to reduce the number of inactive edges and/or words in the sentence which are used to choose the appropriate error rules (see Table 3) or to order these words and edges appropriately. This could be done by guessing where in the input sentence the error is most likely to have occurred. Mellish (1989) suggests that running a top–down parse on an ungrammatical sentence, directly after running a bottom–up parse, can be used to pinpoint the location of an error. This technique was tested on the shortest sentence in our test data (`It working fine now`) with discouraging results: running a top–down parse directly after a bottom–up parse took 697 cycles (90 for the bottom–up parse and 607 for the top–down) compared with 287 cycles for the bottom–up parse followed by our recovery algorithm. This strategy, however, cannot be dismissed before it is tested on a variety of sentences. Again, probabilistic techniques could also play a role in determining the location of an error within a sentence.

6. Conclusions

The robust parsing technique described in this paper is based on the concept of an error grammar, which provides a set of error rules describing ungrammatical sentences in the same way that a conventional grammar provides rules which describe grammatical sentences. The rules in an error grammar are distinguished by means of a rule connective from rules in the normal grammar but they can be parsed using normal chart parsing techniques. The error rules are derived from normal grammar rules on the basis of actual error data, while remaining general enough to ensure good coverage. The recovery algorithm described in this paper is designed to use one error rule together with the results produced by a bottom–up chart parser in order to find a parse for an ungrammatical sentence. The decision to use just one error rule at a time has also been justified by empirical data. The recovery parsing algorithm has achieved promising results in its first evaluation, results which, it is expected, will be improved when the suggestions for further work have been carried out. A further goal is to integrate the work described in this paper with an existing broad–coverage parser, e.g. a probabilistic system or the typed feature structure based LKB system (Copestake, 2002).

Acknowledgements

We are grateful to the reviewers for their very helpful comments. We are grateful to Forbairt Basic Research Grant SC/97/623 and TCD Broad Curriculum Fellowship for funding this research.

Notes

[1] Downloaded as *nlparser.tar.gz* from ftp://ftp.cs.brown.edu/pub/nlparser/ in March 2003.

[2] The error grammar is not a direct reflection of the corpus. It is not learnt automatically from the sentences in the corpus. Rather, an analysis of the types of error occurring in the corpus is carried out and some of the information obtained from this analysis is brought to bear when constructing the error grammar.

[3] There are obvious parallels between this assumption and assumptions in transformational grammar.

[4] We are interested in errors which occur at the sentence level rather than the word level so errors resulting in ill-formed words are not included in the corpus.

[5] It would be worthwhile to complement this research with controlled experiments to judge reliability of 'intended sentence' assessments.

[6] The corpus discussed here is exclusively text based; however, we suspect transcriptions of spoken corpora could be approached effectively in the same manner.

[7] It would be worthwhile to investigate to what extent a broad-coverage probabilistic parser can distinguish between the grammatical and the ungrammatical even though it does not explicitly encode such a distinction. This could be tested by seeing if there is a significant difference between the frequency values obtained by such a parser for ungrammatical sentences and the frequency values obtained for grammatical ones.

[8] It might be interesting to investigate to what extent these kind of error rules provide a treatment for the grammatical phenomenon of ellipsis. That is, the rules of the error grammar may well contribute to the processing of grammatical constructions even with our two-stage process. In terms of Keenan (1976), eliptical sentences are not

'basic'. This is not a strong claim that we make, but a compatible one that is worth exploring.

References

Becker, M., Bredenkamp, A., Crysmann, B. & Klein, J. (1999). Annotation of Error Types for German News Corpus. In *Proceedings of the ATALA workshop on Treebanks*, Paris.

Charniak (2000). A Maximum-Entropy-Inspired Parser. In *Proceedings of the NAACL-2000*.

Copestake, A. (2002). *Implementing Typed Feature Structure Grammars*. CSLI Lecture Notes. Cambridge: Cambridge University Press.

Douglas, S. & Dale, R. (1992). Towards Robust Patr. In *COLING '92*, 468–474.

Earley, J. (1970). An Efficient Context-free Parsing Algorithm. *Commun. ACM* **6**(8): 451–455.

Foster, J. (2000). A Unification Strategy for Parsing Agreement Errors. In Piliére, C. (ed.), *Proceedings of the ESSLLI-2000 Student Session*, 77–87.

Fouvry, F. (2000). Robust Unification for Linguistics. In *ROMAND 2000 1st workshop on Robust Methods in Analysis of Natural language Data,* Lausanne.

Fouvry, F. (2003). Constraint Relaxation with Weighted Feature Structures. In *Proceedings of the 8th International Workshop on Parsing Technologies*, Nancy, France.

Gojenola, K. & Oronoz, M. (2000). Corpus-based Syntactic Error Detection Using Syntactic Patterns. In *NAACL-ANLP00*, *Student Research Workshop,* Seattle.

James, C. (1998). *Errors in Language Learning and Use*: *Exploring Error Analysis*. Addison Wesley Longman.

Jensen, K., Heidorn, G., Miller, L. & Ravin, Y. (1983). Parse Fitting and Prose Fixing: Getting a Hold on Ill-formedness. *Am. J. Comput. Linguist.* **9**(3–4): 147–160.

Keenan, E. L. (1976). Towards a Universal Definition of 'Subject'. In Li, C. (ed.), *Subject and Topic*. London: Academic Press Inc.

Magerman, D. M. & Weir, C. (1992). Efficiency, Robustness and Accuracy in Picky Chart Parsing. In *Proceedings of the 30th ACL*.

Mellish, C. S. (1989). Some Chart-based Techniques for Parsing Ill-formed Input. In *Proceedings of the 27th ACL*, 102–109.

Pereira, F. C. & Shieber, S. M. (1987). *Prolog and Natural-Language Analysis*. CSLI Lecture Notes: Number 10. Center for the Study of Language and Information.

Sampson, G. (2001). Evidence Against the Grammatical/Ungrammatical Distinction. In *Empirical Linguistics*, Chap. 10. Continuum, New York.

Schneider, D. & McCoy, K. (1998). Recognizing Syntactic Errors in the Writing of Second Language Learners. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and the Seventeenth International Conference on Computational Linguistics* (*COLING-ACL*), Vol. 2, Montreal, Canada.

Vogel, C. & Cooper, R. (1995). Robust Chart Parsing with Mildly Inconsistent Feature Structures. In Schöter, A. & Vogel, C. (eds.), *Nonclassical Feature Systems*, Vol. 10. Centre for Cognitive Science, University of Edinburgh, Working Papers in Cognitive Science.

Weischedel, R. M. & Sondheimer, N. K. (1983). Meta-Rules as a Basis for Processing Ill-Formed Input. *Am. J. Comput. Linguist.* **9**(3–4): 161–177.