

Parsing ill-formed text using an error grammar

Jennifer Foster¹

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.

1 Introduction

A traditional rule-based parser, when faced with a sentence which is not described by its grammar, will fail to return any information. 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 tackle this issue. 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

A collection of authentic ungrammatical sentences was analysed. On the basis of this analysis, a set of transformations was applied to a context-free phrase structure grammar yielding an *error grammar* or a grammar of likely ungrammatical sequences. This error grammar contains a sequence of *error rules*, where each error rule corresponds to a particular error which might be expected to occur. When a bottom-up chart parser fails to find a parse for a sentence, a recovery parsing process is undertaken. This recovery phase 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:

A popular approach to robust parsing, the *constraint relaxation* approach, proceeds by repeatedly relaxing constraints in the grammar until a parse for an ungrammatical sentence can be found. [4, 5, 12, 3]. However, 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). Ad-hoc parsing techniques (see for example [9]) are the usual proposed solution to the problem of missing or extra word errors. The approach outlined in this paper does not need to rely on such ad-hoc methods since all errors are dealt with within the uniform framework of an error grammar.

2. A clear model of ungrammaticality:

Probabilistic parsers, such as those described in [2], are by their very nature robust since they make no distinction between the grammatical and the ungrammatical. According to [10], 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 – this means that ill-formed sentences can be diagnosed as such, instead of being viewed merely as sentences occurring with a low frequency. The use of an error grammar is, however, compatible with a probabilistic view of language processing since each individual error rule can be augmented with a probability (derived from a corpus of ill-formed sentences such as a larger version of the one described in section 3), thus allowing one robust parse to be preferred over another.

3. 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 [5] - 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. [3] 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

¹ Computational Linguistics Research Group, Department of Computer Science, Trinity College, Dublin. Email: jfoster@tcd.ie

derived on the basis of empirical linguistic data, and are introduced into the parse in a controlled fashion so that nonsensical parses are never proposed as a solution.

The concept of an error rule is not new. [13], 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 mine but there are differences. The meta-rules are integrated into the main parsing process, whereas in my 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. 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. [11] describe a 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 mine 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.²
2. The parsing process employed by [11] 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 small corpus of syntactic errors was compiled and then used to generate an error grammar from a conventional grammar.

3.1 Error Corpus Collection

A complementary project to this robust parsing study is the compilation of a corpus of ungrammatical language. The corpus currently contains 6,650 words. Every time a sentence containing a syntactic error is noted, the following steps are carried out.

1. The sentence is added to the corpus.

² [7] 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. 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*.

Not only does the error corpus provide us with invaluable information on what kind of errors people tend to make when writing or typing, it also provides us with a set of test data which can be used to test any robust parser. Storing the corrected version of the sentence along with the ill-formed sentence means that the results produced by a robust parser can be easily evaluated. If one of the solutions proposed by the parser matches the corrected sentence, then the parser has successfully parsed the ill-formed sentence. This evaluation procedure is carried out for the robust parsing strategy described here (see Section 5), and it is envisaged that the same procedure could be used to compare the ability of several parsers to understand ungrammatical sentences, irrespective of their approach to parsing.

An important aspect of this corpus compilation is that the context in which the error occurs is always available. This means that the sentences can be corrected without ambiguity. This will not be the case for some ill-formed sentences taken out of context, e.g. in the sentence

Where did these woman learn such arrogance?

taken from the British National Corpus³, there is no way to tell whether the sentence should be corrected by changing the determiner *these* or by changing the noun *woman*.

The errors in the corpus come from the following sources: academic papers and theses, newspapers, magazines, novels, textbooks, websites, lecture notes, student assignments, pamphlets, emails and technical manuals. This is quite a broad source of material, especially when compared to other attempts to create a repository of errors: [6] only collect errors in date expressions in student assignments and newspapers; [1] use an online discussion forum as their only error source.

3.2 Analysing the Errors

Before the errors were analyzed, 50 sentences were randomly extracted from the corpus for the purposes of testing (see section 5). The remaining 306 sentences were analyzed.

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. Fig. 1 indicates the correction operations which were applied to the sentences in the corpus. The frequency of each correction operation is provided along with two examples from the error corpus and their corrections from the parallel corrected corpus.

Another possible correction operation is the *move* operation which moves a word from the sentence into another position within the sentence. However, since such an operation occurs relatively infrequently (2%, according to this corpus), the decision was taken to define it in terms of the add and delete operations, and to treat errors

³ Accessed via <http://sara.natcorp.ox.ac.uk/lookup.html>, April 2000

Correction Operation	Examples
Replace a word (47%)	<ul style="list-style-type: none"> the theory in empirical→the theory is empirical staff was allowed to return→staff were allowed to return
Add a word (27%)	<ul style="list-style-type: none"> 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 (18%)	<ul style="list-style-type: none"> Why is do they appear→Why do they appear Such databases can be to some extent be improved→Such databases can be to some extent improved
More than one of above (8%)	<ul style="list-style-type: none"> 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?

Figure 1. Corpus analysis results

which could be corrected in this way as *composite errors* or errors which can be corrected by applying more than one correction operation. It is also possible to define the replacement operation in terms of the add and delete operations, but since replacement errors occur frequently, this operation is seen as a valid correction operation in its own right.

Another interesting point to note is that 92% of the errors that occur can be corrected by applying just *one* correction operation. It is these errors which will be handled by the robust parsing approach described here. Composite errors are not handled, although they are not incompatible with this approach.

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 1,106 rules was used. The non-terminal symbols of the grammar are augmented with agreement features where appropriate. For each error type, the process of generating error rules for this type is described. The error rules were generated manually but there is no reason why this generation procedure could not be automated.

3.3.1 Replacement errors

In a sentence containing a replacement error, the erroneous word must be replaced by a word that is similar in some way to it. The ways in which two words can be similar correspond to particular types of replacement errors found in the error corpus. These are as follows:

1. 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 one letter in X, or by adding a letter to X, e.g. *nor* with *not*
2. 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. 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. If the word is a preposition, it can be replaced by any other preposition, e.g. *for* with *of*
5. A word can be replaced by a word with the same lexical root but with a different part of speech category, e.g. adjective *syntactic* with adverb *syntactically*

So, 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 right-hand side of this rule, according to the similarity criteria just described. 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 [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 replacement errors which result when the correct and incorrect word are similar in spelling, the connective `agreeop` refers to replacement errors where the correct and incorrect word have conflicting values for an agreement feature, and the connective `vformop` refers to replacement 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 `-- >`.

3.3.2 Missing word errors

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.⁴ So, for example, for the rule

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

the following two error rules will be generated:

```

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

```

For unary rules such as

```
np(Num,Per) --> pro(Num,Per)
```

no error rules with an empty right-hand side are generated. Instead, for each rule which has the category `np(Num,Per)` on its right-hand side, a corresponding error rule is generated which omits this category. An error rule isn't 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)
```

isn't generated since this already exists in the grammar as a conventional rule.

3.3.3 Extra word errors

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 at some position in the rule. This symbol must be capable of matching with any pre-terminal symbol in the grammar: in a typed system it could be the most general word type; in the system described here, a Prolog variable is used, resulting in the added bonus that these error rules can cope with extra constituents as well as extra words. Given, for example, the rule

```
vp_pastp --> v_pastp, pp.
```

the following two error rules are generated:

```

vp_pastp extraop v_pastp, X, pp.
vp_pastp extraop v_pastp,pp,X.

```

⁴ It might be interesting to investigate to what extent missing word error rules provide a treatment for the grammatical phenomenon of ellipsis.

4 Robust Recovery Algorithm

For all *appropriate* error rules

1. Add an active edge corresponding to that rule to the chart agenda
2. Reinvoke the bottom up chart parser - stop when all solutions have been found or when there are no more edges on the agenda
3. If there are solutions record them
4. Remove all edges in the chart which have arisen from the error rule

Figure 2. Recovery Algorithm

The robust recovery parsing algorithm is given in Fig 2. The algorithm takes as input a sentence which has failed to parse using a bottom-up chart parser. 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. All the chart edges built during the normal parsing phase, active and inactive, are available to the recovery algorithm. After an error rule has been tried (whether successfully or unsuccessfully), step 4 wipes the slate clean for the next error rule. It ensures that there is no interaction between error rules.

Fig. 3 details the algorithm which is used to select appropriate error rules. The *errorop* connective corresponds to any connective which appears in an error rule, e.g. *missingop* or *spellop* (see section 3.3).

Step 1 is used to choose error rules corresponding to replacement errors. For this kind of error, appropriate error rules are selected on the basis of the actual words in the input sentence, in a process similar to the chart initialization process.⁵

Step 2 is used to choose error rules corresponding to missing and extra word errors. The chart edge notation

$$W --> \alpha.[start, end]$$

is explained as follows: a category W consisting of the category sequence α has been found between the positions $start$ and end in the input sentence. The selection process is a combination of the bottom-up rule of chart parsing⁶(step 2a) and a left and right scan of the chart (step 2(a)i).

5 Evaluation

The robust recovery algorithm described in Fig. 2 and Fig. 3 was applied to the 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.

⁵ When a chart is initialized, each word in the input sentence is examined and an active edge is added to the chart before this word for each grammar rule which has this word as its right hand side.

⁶ The bottom-up rule of chart parsing states that if there is an inactive edge of a particular syntactic category, X , spanning a particular section of the chart, then any rule which has X as the first category on its right-hand side is added as an active edge at the beginning of this chart section.

1. For all words, w , in the input sentence
 - (a) Add all error rules of the form

$$Z \text{ errorop } w$$
 to the chart in the form of an active edge before w
2. For all inactive edges

$$W \text{ --- } > \alpha.[start, end]$$
 found during the normal parsing phase
 - (a) For all error rules of the form

$$Z \text{ errorop } W \beta$$
 - i. If there is an active edge in the chart ending at position $start$ looking for Z or if position $start$ is at the start of the sentence, and if there is a sequence of inactive edges in the chart starting at position end for the category sequence β , then the rule is appropriate.
 - ii. Otherwise the error rule is discarded.

Figure 3. Algorithm to choose appropriate error rules

5.1 Accuracy

The recovery process was deemed to have worked if one of the corrections it proposed for an input sentence matched the corrected version of the 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:** 2 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. Step 4 in the recovery algorithm means that only one error rule is ever considered during any one parse attempt, with the result that this 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:** 3 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 won't: 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.
3. **Failure to recognize a sentence as ungrammatical:** 3 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**. 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. Short of extending the test grammar so that its grammatical constraints make use of sophisticated semantic and contextual information (in itself a formidable task), these kinds of cases are unavoidable.

5.2 Efficiency

	Average Parse No.	Average Cycle No.
Corrected	2.1	820.4
Ill-formed	4.1	5225.2

Figure 4. Some performance results

Fig.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 [9] 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 the other possible correction, 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 given in Fig.4 is also not a mystery given the large number of error rules. The reduction in the parse cycle number for the ill-formed sentences is a goal for further research. The following are possible ways in which improvements could be made:

1. **Just one correction** The number of parse cycles will be significantly reduced if the recovery phase parsing stops after one error rule has parsed successfully. The problem with this is that, for

some sentences, there is more than one way to correct the sentence and the first correction obtained does not always correspond to the correct one. A probabilistic ordering of the error rules (along the lines proposed by [8]) suggests itself as a possible way around this problem. However, in order for realistic probabilities to be computed, more errors will need to be collected.

2. **Choosing error rules** 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 Fig. 3). This could be done by guessing where in the input sentence the error is most likely to have occurred. [9] 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 needs to be investigated within the context of the approach described here.

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 themselves are realistic since they are derived from normal grammar rules on the basis of actual error data. 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.

ACKNOWLEDGEMENTS

I would like to thank Carl Vogel and the anonymous referees of this paper for their helpful comments and suggestions.

REFERENCES

- [1] Markus Becker, Andrew Bredenkamp, Berthold Crysmann, and Judith Klein, 'Annotation of error types for german news corpus', in *Proceedings of the ATALA workshop on Treebanks*, Paris, (1999).
- [2] Charniak, 'A maximum-entropy-inspired parser', in *Proceedings of the NAACL-2000*, (2000).
- [3] Shona Douglas and R. Dale, 'Towards robust par', in *COLING '92*, pp. 468-474, (1992).
- [4] Frederik Fouvry, 'Robust unification for linguistics', in *ROMAND 2000 1st workshop on Robust Methods in Analysis of Natural language Data*, Lausanne, (October 2000).
- [5] Frederik Fouvry, 'Constraint relaxation with weighted feature structures', in *Proceedings of the 8th International Workshop on Parsing Technologies*, Nancy, France, (23-25 April 2003).
- [6] Koldo Gojenola and Maite Oronoz, 'Corpus-based syntactic error detection using syntactic patterns', in *NAACL-ANLP00, Student Research Workshop*, Seattle, (April 2000).
- [7] Carl James, *Errors in Language Learning and Use: Exploring Error Analysis*, Addison Wesley Longman, 1998.
- [8] David M. Magerman and Carl Weir, 'Efficiency, robustness and accuracy in picky chart parsing', in *Proceedings of the 30th ACL*, (1992).
- [9] Chris S. Mellish, 'Some chart-based techniques for parsing ill-formed input', in *Proceedings of the 27th ACL*, pp. 102-109, (1989).
- [10] Geoffrey Sampson, 'Evidence against the grammatical/ungrammatical distinction', in *Empirical Linguistics*, chapter 10, Continuum, New York, (2001).
- [11] David Schneider and Kathleen McCoy, '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)*, volume 2, Montreal, Canada, (August 1998).
- [12] Carl Vogel and Robin Cooper, 'Robust chart parsing with mildly inconsistent feature structures', in *Nonclassical Feature Systems*, eds., Andreas Schöter and Carl Vogel, Centre for Cognitive Science, University of Edinburgh, Working Papers in Cognitive Science, (January 1995). Volume 10.
- [13] Ralph M. Weischedel and Norman K. Sondheimer, 'Meta-rules as a basis for processing ill-formed input', *American Journal of Computational Linguistics*, 9(3-4), 161-177, (1983).