Robust Parsing with Charts and Relaxation

Peter Ingels
Linköping

Abstract

This paper is a summary of my master's thesis[1] "Error Detection and Error Correction with Chart Parsing and Relaxation in Natural Language Processing" (Ingels 1992). Two methods are presented: The first is a chart-based parsing algorithm inspired by C. Mellish that generates error classifications and, when possible, error corrections to ill-formed input. The algorithm classifies missing words, spurious words, misspellings and substituted words. The second approach presupposes a unification-based grammar formalism. The idea is to extend the PATR formalism so that it can represent alternatives to feature-values. The alternatives can then be used to "carefully" relax constraints imposed by the grammar. Thus the alternatives can be used to abduce corrections in the face of unification failures. The paper also contains a discussion of a proposed project on robustness.

Introduction

NLU-systems that are to be employed in real-world applications need to be able to handle input that violates the expectations of the grammar encoded in them. The occurrences of ungrammatical, or ill-formed, input in such systems is so frequent that it can not be ignored or treated simplistically (e.g. *Sorry, couldn't parse that*).

An informal study of 20 dialogues taken from our own corpus of NLI-dialogues collected with wizard of Oz techniques showed that some 18% of the user utterances contained at least one error. The errors were classified as misspellings, segmentation errors and syntactic errors. It should be noted that the results of the investigation depicted below was collected by a cooperative human, and it is more than reasonable to assume that the number of errors would be higher if an actual system would be used.

66 of the 369 utterances were erroneous (18%)

misspelling	segm. error	synt. error	>1 error/utt.
25	16	21	4

[1]The thesis work was carried out at IRST (Instituto per la Ricerca Scientifica e Tecnologica), Trento, Italy. Oliviero Stock acted as my supervisor. Fabio Pianesi contributed significantly concerning relaxation (see below). I wish to thank them both.

Proceedings of NODALIDA 1993, pages 123-132

The fourth column shows the number of user utterances that contained more than one error.

A system that can handle these and other types of errors is called robust. Robustness can be achieved in different ways, but it requires minimally that the system is able to localize and classify the deviance. There are many different plausible error typologies, the one below is influenced by (Veronis 1991). See also (Stede 1992).

	Performance	Competence
Lexical level	letter substitution letter insertion letter deletion letter transposition	wrong inflection segmentation error grapheme substitution
Syntactic level	word substitution word insertion word deletion word transposition	wrong agreement homophone punctuation error rule violation
Semantic level		presupposition violation reasoning error dialogue law violation conceptual error

Competence errors result from the failure to abide by, or lack of knowledge of, linguistic rules. Performance errors are technical errors made despite knowledge of the rules. The concepts of competence errors and performance errors can of course also be enlarged to encompass errors related to domain knowledge as well as linguistic knowledge.

The appropriate action taken by the robust system in face of ill-formed input is not solely dependent on the error classification. The application in which the system is used is also relevant. Applications range from language tutoring systems over grammar and style-checkers to machine translation and dialogue systems. Spoken communication is also a highly relevant area. So what is to be judged as an appropriate action in an error situation varies.

- The system can enter into a clarification dialogue with the user
- The system can present the user with an error diagnosis
- The system can present the user with a correction hypothesis
- The system can use the best correction hypothesis without bothering the user
- The system can save a partial interpretation of the user utterance
- There might not have been an error (bad coverage)

Two approaches will be presented in the following two sections. First a chart-based technique that can detect constituent errors such as misspelled words (segmentation errors excluded), missing constituents, spurious constituents and substituted words, then a relaxation scheme for detection of constraint violation errors is presented. The relaxation technique has only been partially implemented. The last section is devoted to a discussion on extensions and further research.

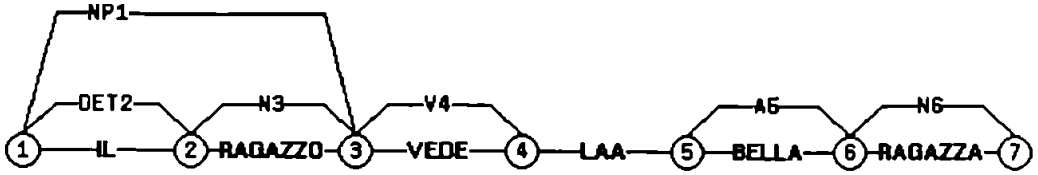
Constituent Errors

The techniques presented in this section rest on Mellish's paper "Some Chart-Based Techniques for Parsing Ill-Formed Input" (Mellish 1989). In his paper Mellish describes a variant of the chart parsing algorithm. His goal is to explore how far detection and classification of errors based purely on syntactic knowledge can lead. Thus he employs a CF-PSG (context-free phrase structure grammar) and the set of standard rules of chart parsing (combination and prediction of edges) is supplemented with a set of error hypothesis rules. These rules can detect and classify missing constituents, spurious constituents and substituted words. Actually he makes misspelling a special case of substituted word!

Mellish's algorithm invites to extensions and alterations and some improvements have also been made to the original algorithm. The improvements basically concerns the error hypothesis rules and some motivations will be accounted for in connection with the introduction of these rules. (There is no room here to present both versions and all considerations taken.)

The generalised chart parsing algorithm basically consists of two phases. First a standard bottom-up parser is supplied with the input. If the bottom-up parser fails the input is in some way ill-formed and recovery is attempted. Then a modified top-down parser is run on the input and the inactive edges left from the bottom-up phase. These inactive edges correspond to the complete constituents found in the bottom-up phase. One of the major differences between the modified top-down parser and the standard top-down parser is that the fundamental rule in the modified parser can incorporate constituents from either direction. In this way the fundamental rule can "narrow down" on an error-point. This scheme calls for a different way to represent an edge's needs and it also affects the top-down rule.

A schematic overview of the basic scheme is given below. The erroneous input in this example is 'Il ragazzo vede laa bella ragazza' ('The boy watches thee pretty girl').



Fgure 1: The chart after the bottom-up phase.

In figure 1, the chart is depicted as it looks when the second phase is ready to start. The superfluous active edges have been "cleaned away" and only the inactive edges that resulted from the bottom-up phase remain. The modified top-down parser would now behave something like:

Hypothesis:	need [S] 1->7
By top-down rule:	need [NP VP] 1->7
By fundamental rule with NP found bottom-up:	need [VP] 3->7
By top-down rule:	need [V NP] 3->7
By fundamental rule with V found bottom-up:	need [NP] 4->7
By top-down rule:	need [DET A N] 4->7
By fundamental rule with A and N found bottom-up:	need [DET] 4->5

This example gives a hint as to what the algorithm does. However, there are further complications. For example, there might be several errors in an input string and hence there must be a way to express multiple needs. If the input string in the example above instead was, 'Il ragazzo vede laa bella ragazza' ('The boy watches thee pretty giirl'), a need like the one below would be useful.

need [DET] 4->5 and [N] 6->7

Furthermore, there are "anchored" and "unanchored" needs. If a couple of consecutive constituents were sought for , say [NP VP] 1->7, and there is neither a complete NP nor a complete VP, this means that there is no way to tell where the two constituents meet. This is expressed with unanchored needs: need [NP] 1->*.

The "*" indicates a vertex in the chart that is not yet determined. Considering all this the general form for an edge will be as follows:

<C S->E needs cl_1 s_1->e_1, cl_2 s_2->e_2, ..., cl_n s_n->e_n>

Where C is a category, the cl_i are lists of categories (which will be shown inside square brackets), S and the s_i are positions in the chart and E and the e_i are positions in the chart or the special symbol "*". An edge of this type in the chart means that the parser is trying to find a constituent of category C, spanning from S to E. In order to do so it must then satisfy all the needs listed (cl_i s_i->e_i).

With this notation the two basic rules, the fundamental rule and the top-down rule, will have the following characteristics:

Top-down rule:
<C S->E needs [c_1,c_2,...,c_n] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>
c_1->RHS (in the grammar)

<c_1 s_1->e needs RHS s_1->e>
Where, if c_2,...,c_n is non-empty or e_1 = * then e = * else e = e_1
Precondition: e_1 = * or c_2,...,c_n is non-empty or there is no edge of category c_1 from s_1 to e_1

Fundamental rule:
<C S->E needs [c_1,...,c_{i-1},c_i,c_{i+1},...,c_n] s_1->e_1, cl_2 s_2->e_2, ...>
<c_i S_1->E_1 needs []>

<C S->E needs [c_1,...,ci_{-1}] s_1->S_1, [c_{i+1},...,c_n] E_1->e_1, cl_2 s_2->e_2, ...>
Precondition: $s_1 \leq S_1$ and (e_1 = * or $E_1 \leq e_1$)

These rules are sufficient to "narrow down" one error like in the example with 'Il ragazzo vede laa bella ragazza'. But since the interest is in the general case, where there can be an arbitrary number of errors in an input string, the parser is expected to by-pass the error-point in some way and to continue to search for possible additional errors. In this way all of an edge's needs will eventually get resolved. This is accomplished by the error hypothesis rules.

Garbage rule:
<C S->E needs [] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>

<C S->E needs cl_2 s_2->e_2, ..., cl_m s_m->e_m>
Precondition: $s_1 \neq e_1$

The garbage rule says that if all constituents of a particular need have been found, and a portion of that need's span is still not covered, this means that this uncovered portion of the chart contains words (or non-words) that should not be included in the parse. The C-constituent spans spurious words/non-words of the input string. That portion of the chart is

consequently disregarded and instead attention is focused on the next need. The garbage rule has not been altered from Mellish's version.

Missing word rule:
<C S->E needs [c_1,c_2,...,c_n] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>

<C S->E needs [c_2,...,c_n] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>
Precondition: c_1 is of lexical category and (s_1 = e_1 or (e_1 = * and (the word at s_1 is not of category c_1 or s_1 = the end of the chart)))

This rule hypothesizes missing word-errors. The rule differs from the corresponding rule in Mellish's algorithm in several respects. He allows for the c_1,c_2,...,c_n to be non-terminals and if s_1 = e_1 he can hypothesize the whole chunk c_1,c_2,...,c_n to be missing. This means that very blunt error classifications are produced, such as e.g. "missing [NP PP]". Furthermore the last clause of the precondition (e_1 = * and (the word at s_1 is not of category c_1 or s_1 = the end of the chart))) is not present in his version. This means that unanchored needs can not have missing constituents, which is an obvious weakness.

Unknown string rule:
<C S->E needs [c_1,c_2,...,c_n] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>

<C S->E needs [c_2,...,c_n] s_1+1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>
Precondition: c_1 is of lexical category and ($s_1 \neq e_1$ or e_1 = *) and s_1 < the end of the chart and the string at s_1 is unknown

Substituted word rule:
<C S->E needs [c_1,c_2,...,c_n] s_1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>

<C S->E needs [c_2,...,c_n] s_1+1->e_1, cl_2 s_2->e_2, ..., cl_m s_m->e_m>
Precondition: c_1 is of lexical category and ($s_1 \neq e_1$ or e_1 = *) and s_1 < the end of the chart and the word at s_1 is not of category c_1

The two last error hypothesis rules have only one counterpart in Mellish's version, namely the unknown word rule. With "unknown" words Mellish means both actual words that do not meet the present expectations and non-words (which obviously do not meet any expectations). With the present rules this distinction is respected. Thus the unknown string rule hypothesizes misspellings and the substituted word rule apply when the input contain a legitimate but misplaced word. However, note that transpositions require that the substituted word rule be applied twice, and so the relationship between the two transposed words is lost.

These extra rules will dramatically increase the parsing search space. In fact the search is exhaustive and obviously the hypothesizing of errors must be controlled in some way. This is done by means of heuristics. For each newly created edge a number of heuristics parameters will be calculated. These scores or penalties will determine an edge's priority compared to other newly created edges. The natural way to realise this procedure is to use the agenda. The agenda will thus be sorted according to the heuristics penalties with the most promising edge in the top position of the agenda. Functions described by Mellish include penalty so far (PSF, edges produced by the error hypothesis rules are penalized), mode of formation (MDE, the formation of unanchored edges are penalized) and several others. See Mellish (1989).

Constraint Violation Errors

This approach relies on the adoption of a feature-based - or unification-based grammar (UBG). The system, that has partially been implemented, makes use of a simple grammar encoded in the PATR-II formalism (Schieber 1986). In this paper the approach is merely sketched. For a full account see (Ingels 1992).

A technique for dealing with constraint violation errors is that of relaxation. This method is addressed in (Douglas & Dale 1992). In the paper D&D approach the problem by stating that some constraints are necessary and others are relaxable. If a unification fails some of the relaxable constraints can be relaxed. If the unification now succeeds a diagnosis of what was wrong with the input can be made. What is meant by relaxing a relaxable constraint in D&D's approach is simply not to incorporate any instantiation of the failed constraint in the resulting FS. In other words, dispose of the failed constraint altogether.

So with a sentence like *Do this cars have a good safety rating?* the resulting feature structure would not have a number feature with D&D&'s approach. A different approach would be to rely on the notion of the conflicting feature values as alternatives, or candidate values. In the example above, parsing *this cars*, the set of candidate values to the unification failure would be singular and plural. In this case other parts of the sentence can provide evidence for a plausible solution to the conflict. The idea is thus to capture the information implied by the unification failure.

The lexicon can also be used to record alternatives. E.g. the ill-formed Italian noun-phrase *la ragazzo* (the boy/girl) can be corrected as il ragazzo (the boy) or as la ragazza (the girl), while *la libro* (the book) only has one plausible correction since there exists no feminine

counterpart to the noun libro. (It should be noted here that alternatives are restricted to atomic values for reasons of complexity.)

The way to implement this scheme would be to explicitly represent the alternatives within the feature structure. So e.g. the Italian definite article *la* would have as value for gender ({f},{f m}), saying that the actual value for the feature gender is feminine although relaxable. The relaxability property is conveyed by the non-empty second component which also explicitly enumerates the possible alternatives to be used in case of unification failure. Non-relaxability is indicated by having the empty set (Ø) as the second component (no alternatives). The Italian noun *libro* could be relaxable having ({m},{m}) as value for gender. The unification (set intersection by pairs) of *la* and *libro* would then produce as value for gender (Ø,{m}), indicating a unification failure (Ø) and the singleton alternative masculine {m}, here functioning as a correction hypothesis.

The natural way to incorporate this scheme with Mellish's algorithm would be to consider only unification proper in the first phase. I.e. do not consider alternatives, look only for well-formed sentences, in the bottom-up phase. Then allow for relaxation in the second, error hypothesizing phase.

A Project on Robustness

A central aspect of the thesis work, presented briefly in the two preceding sections, is that assumptions of error occurrences are made explicitly. In the case of Mellish's algorithm errors are recorded in the chart edges since the error diagnosis is due to the expectations of a particular edge. Also assumptions regarding alternative interpretations of feature structures are explicitly represented. We believe this to be a practicable path to follow in the project too.

To keep track of alternative assumptions/interpretations a reasoned chart parser will be used. For a good survey of reasoned chart parsers see (Wirén 1992). A reasoned chart parser is a chart parser where dependencies between edges are explicitly recorded. With this framework the likelihood of alternative interpretations can be judged with reference to the assumptions on which they rest. In his dissertation Wirén suggests the reasoned chart parser to be integrated with an ATMS-based problem solver to support also such assumptions that can not be represented as chart edges. This setting will be used in the project as a general formal framework for studying diagnosis and interpretation of ill-formed input. Alongside with this we will gain knowledge of the error types occurring and their relative frequency. Another thing that should be empirically

investigated is the question regarding what action is appropriate in different error situations. This includes preventive actions.

The empirically collected information will then, together with the formal framework, serve as a basis for an implementation of a robust and reasonably fast interpreter, eventually to be integrated in the BILDATA-system. The BILDATA-system is the (written) dialogue system in our current project 'Dynamic Natural Language Understanding' (Jönsson 1993) .

Some of the questions relating to the implementation resulted from the thesis work.

Although my version of Mellish's algorithm makes the error classification more fine-grained than Mellish himself does, the error classification is inadequate. Transpositions and segmentation errors e.g. can not be dealt with in a straight forward manner. The reason being that an error hypothesis is kept local in an edge. That is not a problem as long as errors are discovered incrementally, one at a time, but when several constituents or input fragments are affected by a single error, there is a problem. This also raises the question whether there are any profitable alternatives to the two stage process suggested by Mellish. Maybe one should look out for 'lower level errors' (segmentation errors, misspellings,...) already in the first phase or in a third intermediate phase?

When should the system give up trying to parse the ill-formed input? Presently the system can parse everything (, you can put your elbow on the keyboard and the system will eventually come up with a diagnosis of what went wrong). The subtle question reads: how distorted can an utterance be and yet be understandable? What are the criteria for stating that the input is simply rubbish?

Another problem is the systems inability to discriminate between competing correction hypotheses. One reason for this is obviously that the system uses only syntactic information in the diagnosis process. Should semantic constraints be an integral part of the grammar and used as a filter in the parsing process?

References

Ingels, Peter. 1992. *Error detection and error correction with chart parsing and relaxation in natural language processing*. Master's Thesis, Department of Computer and Information Science, Linköping University, LiTH-IDA-Ex-9269.

Jönsson, Arne. 1993. *Dialogue Management for Natural Language Interfaces — An Empirical Approach.* Linköping Studies in Science and Technology, Dissertation No. 312.

Douglas, Shona and Robert Dale. 1992. *Towards Robust PATR.* In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-92)*, Vol. II: 468–474.

Mellish, C. S. 1989. *Some Chart-Based Techniques for Parsing Ill-Formed Input.* In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pp. 102–109.

Schieber, Stuart M. 1986. *An Introduction to Unification-based Approaches to Grammar.* P. University of Chicago Press, Chicago, Illinois, USA.

Stede, Manfred. 1992. *The Search for Robustness in Natural Language Understanding.* ARTIFICIAL INTELLIGENCE REVIEW, Vol. 6 No. 4: 383–414.

Véronis, Jean. 1991. *Error in Natural Language Dialogue Between Man and Machine.* INTERNATIONAL JOURNAL OF MAN-MACHINE STUDIES, Vol. 35, No. 2.

Wirén, Mats. 1992. *Studies in Incremental Natural-Language Analysis.* Department of Computer Science, Linköping University. Linköping Studies in Science and Technology, Dissertation No. 292.