

# Hierarchical Multiple Error Recovery Based on Chart Parsing

Kyongho Min & William H. Wilson

School of Computer Science & Engineering

The University of New South Wales

Sydney 2052 Australia

{min,billw}@cse.unsw.edu.au

## Abstract

This paper describes a system for detecting and correcting multiple, non-adjacent, syntactic errors using only syntactic information. The system employs two-stage error recovery using well-formed fragments left behind by a chart-based parser, and a context-free grammar. We focus on the problem of dealing with sentences with more than one error, while not reducing performance on single-error sentences. The usefulness of head-driven parsing and seed parsing (cf. island parsing) for multiple error recovery is discussed.

## 1 Introduction

Humans often make and encounter errors when communicating in a natural language. The errors may be no more than typographical or they might arise from incomplete command of the language: either can cause significant problems for natural language processing systems (Eastman & McLean, 1981).

Past work in this area has addressed varying aspects of automatic detection and correction of errors at the typographical (Damerau & Mays, 1989), morpho-syntactic (Vosse, 1992), syntactic (Mellish, 1989; Kato, 1994), and semantic levels (Fass & Wilks, 1983). These have been based on a range of syntactic formalisms and algorithms, including ATNs (Weischedel & Sondheimer, 1983), augmented context-free grammar (Vosse, 1992), and chart parsing with a context-free grammar (Mellish, 1989; Kato, 1994). Most base their schemes on the assumption that a single system handles both single and multiple error sentences. Here we focus on multiple-error sentences, while not reducing performance on single-error sentences.

This paper describes a system for repairing multiple syntactic errors, provided the errors do not occur in adjacent positions in a local tree. The system, called MERCHANT (Multiple Error Recovery with CHARTs and Need-arc Trees), uses the well-formed fragments left behind by a chart-based parser using a context-free grammar. Our system adopts many features of Mellish's system (1989). Mellish's system was controlled with six rules and a scoring scheme using six parameters to select a rule for the best correction of an error. However, the scoring scheme makes multiple error correction fairly complex.

Following Mellish's system, Kato (1994) improved the chart-based error recovery system by employing bidirectional parsing for the fundamental rule. Kato applied two parameters to decide the best rule to detect an error; the number of errors corrected so far and the number of constituents needed.

MERCHANT differs from the above systems as follows. First, in MERCHANT the best repair depends on the type of error and its importance in the local tree (e.g. a head daughter versus a non-head daughter in a local tree) rather than on a rule selection scheme.

Second, MERCHANT has three recovery stages; a parser for well-formed sentences, a second parser to recover from a single error, and a third parser to detect and correct multiple errors. Thus each subsystem focuses on a particular class of error.

Third, Mellish's system used a dangling position symbol, \*, to denote a position to be determined by further rule application (e.g. NP needed from 0 to \*, VP needed from \* to 7). We avoid the use of a dangling position. Though the dangling position can be useful for detecting and correcting multiple errors, in the worst case, its determination is delayed until error correction. By defining the position of needed constituents (e.g. NP or VP) early, local errors can be effectively localised and detected.

In multiple error recovery, three factors will be discussed along with the effectiveness for multiple error recovery of two general parsing schemes: head-driven parsing and the seed parsing rule. The three factors are: (1) how to detect each error type, (2) how to determine the number of errors by localising each error, and (3) how to select the best repair among many repairs at the syntactic level.

The above factors will be discussed in sections 2 and 3. Section 4 describes experiments and their results. Section 5 describes strengths and problems.

## 2 Methods used in Multiple Error Recovery

The MERCHANT system is based on a single-error recovery system with two stages (Min & Wilson, 1995): a basic bottom-up chart parser, WFSCP (Well-Formed Sentence Chart Parser). If WFSCP did not succeed (within the scope of the grammar/lexicon), a second parser, termed IFSCP (Ill-Formed Sentence Chart Parser), was invoked to try to repair the sentence. If IFSCP fails to recover

from a single error, then MERCHANT, a parser for multiple error recovery, is invoked. Since MERCHANT extends this previous work, the present paper focuses on the requirements and effects of multiple error recovery with an *assumption* that multiple errors are not adjacent.

MERCHANT detects multiple errors by localising each error using unsatisfied goals from the single error recovery system (IFSCP). Each unsatisfied goal becomes an *mgoal* (a goal for multiple error recovery - it includes the constituents needed for error correction together with the two positions (from, to) where it is needed). When IFSCP attempts a single error recovery, if either of (NP PP) from *i* to *j* could not be found because of multiple errors, then an *mgoal*, (NP PP) from *i* to *j*, is generated. After collecting all *mgoals* from IFSCP, MERCHANT is invoked to localise each error using two rules: the *head-driven parsing rule* (cf. Proudian & Pollard, 1985) and the *seed-parsing rule* (cf. island parsing).

The application of these rules is similar to bidirectional parsing (Satta & Stock, 1994) except for considering possible errors; we consider them in either direction for both head and seed constituents. Exactly one of two rules is applied depending on the *mgoal*'s characteristics. If there are seeds for needed constituents of an *mgoal*, then the seed-parsing rule is applied. If no seed of needed constituents is found, then the head-driven parsing rule is used. A *seed* for an *mgoal* is an inactive arc found by WFSCP and within the scope of the *mgoal*. For example, if there is a goal with needed constituents (DET ADJP NOUN) from 0 to 3, and the initial parse had found the ADJP, then that ADJP would be a '*seed*'. Thus the seeds of a local tree are the known constituents, viewed as starting points for recovery.

The position of each needed constituent in the *mgoal* is used to localise a single error. It is convenient to define a function  $\text{Pos}(C_i, (C_1 \dots C_n))$  which returns the position (left, middle, right, or leftright) of  $C_i$  ( $1 \leq i \leq n$ ) in the list of constituents. Thus  $\text{Pos}(C_1, (C_1 \dots C_n)) = \text{left}$ , and  $\text{Pos}(C_n, (C_1 \dots C_n)) = \text{right}$ .  $\text{Pos}(C_i, (C_1 \dots C_n)) = \text{middle}$  if  $1 < i < n$ .  $\text{Pos}(C_1, (C_1)) = \text{leftright}$ .

The function  $\text{Pos}$  is applied to needed constituents of an *mgoal* to determine what constituents to expect adjacent to known constituents (both left and right) of a particular constituent. If the rightmost constituent of an NP  $\rightarrow$  DET NOUN is known, then the fact that  $\text{Pos}(\text{NOUN}, (\text{DET NOUN})) = \text{right}$  leads to the expectation that it will be possible to determine the lexical category of left neighbour of the NOUN (but not necessarily the right neighbour).

If (NP VP) is an *mgoal*, and the rightmost constituent of NP is known (say it's a NOUN), then

at least the right constituent of the *mgoal* (VP) can be expected to the right of the NOUN of the NP.

## 2.1 Head-Driven Parsing Rule

The head-driven parsing rule uses head daughters of a particular phrasal constituent in an *mgoal* to search for seeds covering the head daughter constituents (Fig. 1). In right-corner head rule in the diagram in figure 1, the *mgoal* is  $(C_1 C_2)$ . If a head daughter (e.g. H in the figure) is located in the rightmost position of  $C_1$ , then this rule is applied using the seeds found. In terms of the function  $\text{Pos}$ , the right-corner head rule is applied if and only if

- $C_1$  is a phrasal constituent,
- $\text{Pos}(C_1, (C_1 C_2)) = \text{left}$  or *leftright*,
- $\text{Pos}(H, (r_1 r_2 H)) = \text{right}$  in a context free rule  $C_1 \rightarrow (r_1 r_2 H)$  - H is a head daughter of  $C_1$  -, and seeds for H are found.

**(a) Head-Driven Parsing Rule**

- assume *mgoal*  $(C_1 \dots C_n)$  is needed from *j* to *k*
- for** each  $C_i$  that is not already found and is a phrasal category **do**
- if**  $\text{Pos}(C_i, (C_1 \dots C_n)) = \text{left}$  or *leftright*,
- then do** right-corner head rule with  $C_i$
- if**  $\text{Pos}(C_i, (C_1 \dots C_n)) = \text{right}$  or *leftright*,
- then do** left-corner head rule with  $C_i$

**(i) Right-Corner Head Rule** with  $C_i$  (from the *mgoal*  $(C_1 \dots C_n)$  from *j* to *k*)

Find the head-daughter types  $\{H_1 \dots H_n\}$  of  $C_i$  (i.e., heads of RHSs of grammar rules for  $C_i$ );

For each  $H_p$ , find seeds  $(S_{p1} S_{p2} \dots S_{pm(p)})$  lying between *j* and *k* -  $(\sum_{r \in \{i+1 \dots n\}} \text{MEL}(C_r))^*$

**if**  $H_p$  is the rightmost constituent in the RHS of some grammar rule for  $C_i$ ,

**then** make two localised single-error goals using each seed  $\{S_{pq}\}$  of  $H_p$

- goal  $C_i$  from *j* to *end of*  $S_{pq}$
- goal  $(C_{i+1} \dots C_n)$  from *end of*  $S_{pq}$  to *k*

Ignore other cases;

\*mel (minimal extension length) means the minimal number of preterminal categories which are necessary for the production of the rule's LHS category. For example, the mel of S is 2, because of examples like *I go*.

Right-Corner *mgoal*  $(C_1 C_2)$   
Head Rule

**(ii) Left-Corner Head Rule**

Analogous to right-corner head rule.

Figure 1. Head-Driven Parsing Rule

For example, there is an ill-formed sentence with multiple errors, *Teh (The) telephone is by they (the) window*. An mgoal (NP VP) from 0 to 6 would be produced. In this case, NP is a phrasal constituent and Pos (NP, (NP VP)) = 'left'. Thus the system applies the right-corner head rule. Possible head daughters of NP include NOUN and PRON. NP → DET NOUN is a likely rule: Pos (NOUN, (DET NOUN)) is 'right'. If seeds for NOUN are found (e.g. (NOUN "telephone" from 1 to 2)), then single error goals are produced by using both needed constituents of the mgoal and found seeds. One goal for a single error is (NP) from 0 (the starting position of the mgoal) to 2 (the ending position of the found seed NOUN) and the other is (VP) from 2 to 6 (the ending position of the mgoal).

## 2.2 Seed Parsing Rule

The seed parsing rule is similar to bidirectional island parsing (Fig. 2). However, the seed parsing rule takes into account possible errors in either side of a seed (cf. island). In applying the seed parsing rule, the seeds are searched for by using each needed constituent of an mgoal rather than head daughters of each needed constituent. In this sense, the seed parsing rule is similar to island parsing (Satta & Stock, 1994).

Each of three seed parsing rules is applied depending on the position of each constituent in the mgoal. Suppose there is an mgoal ( $C_1 \dots C_n$ ) from  $j$  to  $k$ . The seeds ( $S_i$ ) for  $C_i$  ( $1 \leq i \leq n$ ) - not head daughters of  $C_i$  in the head-driven parsing rule - are sought between  $j$  and  $k$ . If Pos( $C_i, (C_1 \dots C_n)$ ) is 'left', (i.e.  $i=1$ ) then a goal for a single error can be made with needed constituents ( $C_2 \dots C_n$ ) and found seeds ( $S_i$ ). For example, if there is an mgoal (DET ADJP NOUN) from 2 to 7 for the sentence *They saw er (an extra word) a vrey (very) big dinosaur*. Pos (DET, (DET ADJP NOUN)) is left and a seed for DET (e.g. DET "a" from 3 to 4) is found. Thus single error goals are produced: (ADJP NOUN) is needed from 4 (the end of the seed DET) to 7 (the end of the mgoal); and (DET) is needed from 2 (the start of the mgoal) to 4. The other two seed rules apply in a similar way (see figure 2).

The aim of applying both rules to mgoals is effectively to suggest a goal which may include a single error in order to reduce possibility of combinatorial explosion while detecting and correcting multiple errors. When an input sentence has two unknown words, the possible combinations are 2 (substitutions) x 2 (deletions).

If a single error goal is the same as a previous single error goal, then the goal is discarded to avoid redundancy of error recovery. After localizing a single error for each mgoal, MERCHANT invokes a single error recovery scheme which is basically the

same as the single error recovery scheme of Min & Wilson (1995).

**(b) Seed Parsing Rule**

- the mgoal ( $C_1 \dots C_n$ ) is needed from  $j$  to  $k$
- for** each  $C_i$  that is not already found **do**
- if** Pos ( $C_i, (C_1 \dots C_n)$ ) = *left* ( $i = 1$ ),
- then do** left-corner seed rule
- if** Pos ( $C_i, (C_1 \dots C_n)$ ) = *middle* ( $1 < i < n$ ),
- then do** middle seed rule
- if** Pos ( $C_i, (C_1 \dots C_n)$ ) = *right* ( $i = n$ ),
- then do** right-corner seed rule

**(i) Left-Corner Seed Rule**

- find seeds ( $S_1 \dots S_m$ ) of  $C_i$  in the mgoal ( $C_1 \dots C_n$ ) from  $j$  to  $k - (\sum_{r \in \{i+1 \dots n\}} MEL(C_r))$
- make localised single-error goals using each seed  $S_p$  of  $C_i$ 
  - goal  $C_i$  from  $j$  to *end of*  $S_p$
  - goal ( $C_{i+1} \dots C_n$ ) from *end of*  $S_p$  to  $k$

**(ii) Middle Seed Rule**

- find seeds ( $S_1 \dots S_m$ ) of  $C_i$  in the mgoal ( $C_1 \dots C_n$ ) from  $j + (\sum_{r \in \{1 \dots i-1\}} MEL(C_r))$  to  $k - (\sum_{r \in \{i+1 \dots n\}} MEL(C_r))$
- make localised single-error goals using each seed  $S_p$  of  $C_i$ .
  - goal ( $C_1 \dots C_{i-1}$ ) from  $j$  to *start of*  $S_p$
  - goal ( $C_{i+1} \dots C_n$ ) from *end of*  $S_p$  to  $k$

**(iii) Right-Corner Seed Rule**

- find seeds ( $S_1 \dots S_m$ ) of  $C_i$  in the mgoal ( $C_1 \dots C_n$ ) from  $j + (\sum_{r \in \{1 \dots i-1\}} MEL(C_r))$  to  $k$
- make localised single-error goals using each seed  $S_p$  of  $C_i$ 
  - goal ( $C_1 \dots C_{i-1}$ ) from  $j$  to *start of*  $S_p$
  - goal  $C_i$  from *start of*  $S_p$  to  $k$

Figure 2. Seed Parsing Rule

## 3 Error Correction and Scoring Scheme

The single error goal is processed by a general top-down parsing strategy incorporating top-down expectation and bottom-up satisfaction in order to detect and correct an error. A single error is detected using two data structures, goals and need-arcs containing information about which constituents are found and which are needed between two positions (*from to*) in the string (see Min & Wilson, 1995). After an error is detected, first the bottommost (pre-terminal) errors are corrected, in two phases; from a goal or a need-arc. If a goal satisfies a correction condition (e.g. substitution error, deletion error), the goal is transformed to a correct constituent using the goal's information. Suppose a goal, (NOUN) is needed from  $i$  to  $i+1$ , is generated. The goal satisfies substitution condition, and a new constituent, (NOUN) from  $i$  to  $i+1$ , is produced.

At this stage, an integrated spelling corrector (Min & Wilson, 1995) is invoked to detect and correct any spelling error, as a substitution error may be caused by a spelling error. Our spelling corrector is based on Damerau's study (1964). Thus a single character error can be handled by a dictionary lookup strategy.

When a word-deletion error occurs, a goal, (e.g. ADJ needed from *i* to *i*) is generated. The goal satisfies the addition condition, and a new constituent, (say ADJ from *i* to *i*), is produced. When an extra word is added, the correction should delete the extra known or unknown word. The error correction (deletion) is made using a need-arc. If all constituents for a local tree are found and an extra word is left, then the useless extra word can be deleted (cf. Mellish's garbage rule).

After all bottommost errors are corrected, the corrections are used to reconstruct the first goal node (S). In our system, the derivational history of a particular correction is provided by a hierarchical network, termed the need-chart network (Min & Wilson, 1995). The need-chart network links every need-arc produced by both IFSCP and MERCHANT hierarchically. The need-chart network makes reconstruction of the S-node fast once the bottom-level corrections are done.

MERCHANT uses two criteria to select the best repair among alternatives; (1) error types and (2) heuristics for importance of a repaired constituent in a local tree. By error types, a substitution error is given less penalty than an addition or a deletion error. By weight in a local tree, if the repaired constituent is a head daughter, the local tree is given less penalty than that of a non-head daughter. The weight penalty is accumulated as the network is retraced up to the S node. Thus the depth of the bottommost error correction affects the total penalty for the S node.

#### 4 Results of Experiments

We randomly selected 15 sentences from Oxford Advanced Learner's Dictionary with three different lengths (6, 9, 12) and used them as a basis for a test set, as described below, with 96 context-free grammar rules (which can handle declarative sentences, inversion, WH-question, and passive voice, but not embedded relative clauses and which do not consider syntactic information such as number, case, tense, and verb subcategorisation). When parsing these base sentences, the parser produced a single parse tree for each.

Each sample sentence is transformed to an ill-formed sentence with a single error, double errors, and triple errors with five types of error (replacement of unknown/known words, addition of unknown/known words, and deletion of words).

With double errors, each sentence is transformed to an ill-formed sentence with two errors randomly, but the errors are sub-classified into two types; an error introduced by head constituents (in fact NOUN and VERB only, for testing, rather than general concept of head constituents) and non-head constituents (e.g. DET etc.).

After producing double-error sentences, one of errors is corrected to produce single-error sentences with an error of either a head constituent or non-head constituent. Thus, we tested 75 sentences containing a single error, 150 sentences with double-error, and 75 sentences with triple-error. In the experiment, the combinations of different error types are not considered. We implemented and tested our system using Macintosh Common Lisp 2.0 with 10MB working memory on a Macintosh IIfx.

Table 1 shows that the single error recovery system spends 2-4 times as long in parsing time and 5-15 times as many (repaired) parse trees, compared to parsing a sentence without an error. In terms of types of error correction, substitution corrections are more common than the others, as spelling corrections are involved with substitution errors. If there are five alternative spelling corrections, then five S trees are produced in substitution corrections. When a word (either unknown or known) is added, the system gives the best correction performance (77% of 15 testing sentences). The worst case is deletion errors: the first or second best correction of a deletion error was correct in 27% of 15 testing sentences. This is because most of the errors detected were classed as substitutions, not deletions: a substitution error gets less penalty than other two types of error.

Table 2 shows the complexity of recovering from multiple errors and the difficulty of selecting the best correction among alternative repairs. The column 'error correct' represents the reconstruction of S tree rather than the correction of local errors. For example, a sentence including triple errors, *Teh (The) lamp it (is) on the tavle (table)*, was not correctly repaired. However, our system based on both parsing rules detected and corrected 2 local errors (e.g. (DET *teh*), (CARDINAL *ten*) for the word *Teh* and (NOUN *tale, table*) for the word *tavle*) and reconstructed phrasal constituents (e.g. (PP *on the tale*), (PP *on the table*), (NP *the lamp*) (NP *ten lamp* - no number information)). When localising a single error, two errors are well localized except for the error for (BE *it*).

When employing hierarchical approaches for multiple error recovery, the time cost for the single error recovery system (IFSCP<sub>S</sub>) is 1.4% to 5% of that of multiple error recovery system (IFSCP<sub>M</sub>). Among five types of errors, substitution errors (either known or unknown) introduced by head

Error Types	No. of Error	Time (sec)	No. of Sols	Correct.	Sols.	Ranks	Error	correct.	Type
				1st - 2nd	3rd - 5th	> 5th	subst	delete	add
No error	0	1.2	1	-	-	-	-	-	-
subst unknown word	1	3.1	3.9	8/15	7/15	0	3.6	0.3	-
add unknown word	1	2.7	5.2	13/15	2/15	0	3.8	1.4	-
subst known word	1	2.8	5.1	6/15	5/15	4/15	3.9	0.4	0.8
add known word	1	3.2	9.7	10/15	4/15	1/15	5.7	1.5	2.5
delete a word	1	4.7	15.3	4/15	5/15	6/15	9.9	0.9	4.5

**Table 1.** Result of Single Error Recovery (Average of 15 sentences)

Error Type	No of Errors	Error Word	Error correct	Time (sec)		No. of Sols	Sols in 1st - 5th	Error correct		Type
				IFSCPS	IFSCPM			subst	delete	
subst unknown words	2	no-head	14/15	1.0	20.8	10	8/15	16	4	-
	2	head	12/15	0.3	21.9	9	12/15	18	-	-
	3	-	0/15	0.4	17.8	0	0/15	-	-	-
add unknown words	2	no-head	9/15	0.9	40.8	4	6/15	5	3	-
	2	head	13/15	1.0	40.4	7	6/15	9	5	-
	3	-	0/15	0.6	47.3	0	0/15	-	-	-
subst known words	2	no-head	13/15	1.6	47.6	21	4/15	28	5	9
	2	head	13/15	0.4	22.5	19	9/15	30	2	6
	* 3	-	5/15	0.5	33.2	7	0/15	9	1	4
add known words	2	no-head	12/15	6.1	181.4	12	0/15	10	5	9
	2	head	14/15	6.7	336.9	19	4/15	20	6	12
	* 3	-	4/15	7.6	617.1	26	0/15	36	9	7
delete words	2	no-head	12/15	3.6	111.0	57	2/15	48	3	63
	2	head	15/15	1.3	93.5	33	3/15	47	1	18
	* 3	-	4/15	1.3	98.1	51	0/15	53	1	48

**Table 2.** Result of Multiple Error Recovery (Average of 15 sentences)

\*-row means that the triple errors are detected and corrected as double-error. All values are average of the number of double-error detection and correction except for parsing time column. In *error word* column, head = NOUN or VERB and no-head = DET, PREP, ADJ, ADV, etc.

constituents shows the best result in terms of the best correction measured by the first five solutions of alternatives. When the head constituent is involved in errors, the best correction rate in the first five alternative repairs is better than that of non-head constituents.

If two known-word errors comprising substituting head constituents should occur, then the parsing cost is half of those of non-head constituents. However, if two known-word errors comprising adding head constituents occur, then the parsing cost is double of those of non-head constituents. In the case of unknown word errors, the parsing cost between head constituent errors and non-head constituent errors is very similar.

The error recovery from two errors takes 7-81 times more parsing time than that from single

errors. In the case of an addition error, the time cost of a double error can be 81 times that for a single error.

## 5 Discussion and Conclusion

### 5.1 Discussion of the Results

We employed 96 context-free grammar without using number, case, tense, and verb subcategorization information. This, and the spelling corrector, led to an explosion in the number of solutions. If a richer grammar were used, then the number of solutions ought to be greatly reduced. If more syntactic information for spelling correction were provided, then the problem with the spelling correction would be reduced. The spelling corrector usually did well with longer words.

In the case of double errors, the seed parsing rule and head-driven parsing rule succeed in localizing a single error (in 85% of testing sentences). However, the system fails to localize single errors when there are triple errors in a sentence and detects triple errors as double. In this case, both rules may be recursively applied to localise each error. If the localized error is recognised as multiple errors by the error-correction process, then both the seed parsing rule and head-driven parsing rule can be applied to the localised error again. This further refinement of local errors takes more time and suggests more repairs than the current system according to our preliminary testing. Even though the current system fails to reconstruct S repairs with 15% of double errors and finds no repairs for triple errors, we prefer the current system because cases of triple errors in a sentence are very rare.

If a goal for a multiple error fails to localize a single error by both rules, then the current system ignored the goal. For further improvement, a non-head-driven parsing rule will be employed for goals not handled by current rules. In this case, the leftmost or rightmost constituent of a rule's RHS will be considered as a seed for a single error localisation.

The Trains 91 Dialogues from the University of Rochester (Gross, Allen, & Traum, 1993) is a transcription of spoken conversations in the Trains domain. The conversations include lots of multiple errors. We will test the competence of our system using the real data for further improvement.

## 5.2 Conclusion

We described hierarchical multiple error recovery system employing both seed parsing rule (e.g. island parsing rule) and head-driven parsing rule to recover from multiple errors. MERCHANT detected and corrected 85% of testing sentences with double errors by successfully reconstructing S node.

The hierarchical strategy paid off in terms of extra time cost (2.6% of multiple error recovery system for running a single error recovery system). It is also possible for the hierarchical system to detect and correct double errors as a single errors. However, if the input sentence includes a single error, then the second parser focused on single error recovery efficiently detects and corrects a single error.

MERCHANT's use of two rules effectively reduced the combinatorial problem in multiple error detection and correction without reducing the power of a single error correction. This is important given the rarity of multiple errors in a

sentence compared to single errors, in many categories of real-life data.

## Acknowledgment

We would like to thank Dr. John Shepherd for useful discussion on this work and helpful comments on earlier drafts of this paper.

## References

- Damerau, F. & Mays, E. (1989). An Examination of Undetected Typing Errors. *Information Processing & Management*, **25**(6), 659-664.
- Eastman, C. & McLean, D. (1981). On the Need for Parsing Ill-formed Input. *AJCL*, **7**(4), 257.
- Fass, D. & Wilks Y. (1983). Preference Semantics, Ill-formedness, and Metaphor. *AJCL*, **9**(3-4), 178-187.
- Gross, D., Allen, J. & Traum, D. (1993). The Trains 91 Dialogues. Technical Note 92-1, Computer Science Department, The University of Rochester.
- Kato, T. (1994). Yet Another Chart-Based Technique for Parsing Ill-Formed Input. Fourth Conference on Applied Natural Language Processing, 107-112.
- Mellish, C. (1989). Some Chart-based Techniques for Parsing Ill-formed Input. *ACL Proceedings, 27th Annual Meeting*, 102-109.
- Min, K & Wilson, W. (1995). Syntactic Recovery and Spelling correction of Ill-formed Sentences. *Abstracts of the Third Conference of the Australasian Cognitive Science Society*, 81. Full version to appear in *Selected Papers from the 1995 Australasian Cognitive Science Society Conference*.
- Proudian, D. & Pollard, C. (1985). Parsing Head-Driven Phrase Structure Grammar. *ACL Proceedings, 23rd Annual Meeting*, 167-171.
- Satta, G. & Stock, O. (1994). Bidirectional Context-free Grammar Parsing for Natural Language Processing. *AI*, **69**, 123-164.
- Vosse, T. (1992). Detecting and Correcting Morpho-Syntactic Errors in Real Texts. *ACL Proceedings, Third Conference on Applied Natural Language Processing*, 111-118.
- Weischedel, R. & Sondheimer, N. (1983). Meta-rules as a Basis for Processing Ill-formed Input. *AJCL*, **9**(3-4), 161-177.