

Syntactic Recovery and Spelling Correction of Ill-formed Sentences

Kyongho Min and William H. Wilson
School of Computer Science & Engineering
The University of New South Wales
SYDNEY NSW 2052 Australia
Email: {min,billw}@cse.unsw.edu.au

Abstract

This paper describes syntactic repair and spelling correction of ill-formed sentences within a context-free grammar using non-static filtering, of ill-formed sentences which violate subject-verb agreement or premodifier-noun agreement. The system described here provides recovery of local trees, reconstruction of the sentence, and spelling correction of detected typographical errors. It also produces a report on the repair that has been carried out. The system includes generalised problem-solving strategies for detecting and correcting several types of syntactic and typographical errors, and also includes heuristics.

The paper focuses on a system for the integration of lexical and syntactic recovery of ill-formed sentences at the local tree and the final goal (sentence) level. The system is based on a chart parser and employs a mixed top-down/bottom-up strategy together with left-to-right and right-to-left parsing. The implementation is composed of two chart parsers: a well-formed sentence chart parser (WFSCP) and an ill-formed sentence chart parser (IFSCP), and a spelling correction algorithm based on dictionary lookup.

Keywords: ill-formed text, robust parsing, chart parsing, spelling correction

1 Introduction

Human beings often produce sentences that are ill-formed at various levels, including the typographical/morphological, syntactic, semantic, and pragmatic levels. When we encounter ill-formed sentences we usually understand their meanings and tolerate the errors. The intended meaning can often be inferred by using a variety of linguistic and real world knowledge.

Shanon (1973) tested the interpretation of spoken ungrammatical sentences in Hebrew, with one to three errors in a sentence. The errors included violation of agreement rules (e.g. number, gender, and tense). Shanon found that humans preferred particular types of correction, for example in number-agreement violation, the verb is replaced rather than the noun, and in tense-agreement between a verb and an adverbial phrase, the verb is replaced rather than the adverbial phrase. He also found that unmarked forms such as verbs in infinitive form are replaced more than inflected forms. Shanon also observed a least effort principle: if there are two possible changes - for example a single change or a double change -, then the simpler change is preferred.

An experiment by Cooper, Tye-Murray, and Nelson (1987) explored humans' tolerance of ill-formed sentences. Their results, based on spoken text which included missing words showed that listeners could detect errors better when they paid attention to detecting errors than when they didn't pay attention to this; listeners were highly accurate in reporting the presence of missing words (96 % of 190 cases) when they were forewarned about the specific types of errors that might be encountered while listening. But the detection of missing words (34% of 80 cases) was quite poor when the words were easily predictable from context and when listeners were not forewarned about the specific types of errors.

This paper focuses on the detection and correction of ill-formed sentences including a single syntactic error introduced by replacement of a valid word by a known/unknown word, insertion of an extra known/unknown word, or by deletion of a word. Many systems have focussed on the recovery of ill-formed texts at the typographical level (Damerau, 1964; Damerau & Mays, 1989), the morpho-syntactic level (Vosse, 1992), the syntactic level (Hayes & Mouradian, 1981; Mellish, 1989), the semantic level (Fass & Wilks, 1983), and the pragmatic level (Granger, 1983). Those systems described how to identify a localised error and how to repair it

using grammar independent rules (Mellish, 1989), grammar specific rules (meta-rules) (Weischedel & Sondheimer, 1983), semantic preferences (Fass & Wilks, 1983), and heuristic approaches (Damerau, 1964; Damerau & Mays, 1989; Vosse, 1992).

Weischedel & Sondheimer (1983) used the term *one-stage error recovery* for a system that can process both ill-formed and well-formed sentences with a single parser. Our system is a *two-stage error recovery* parser based on chart parsing and consists of a well-formed sentence chart parser (*WFSCP*) and an ill-formed sentence chart parser (*IFSCP*) with a spelling correction algorithm based on dictionary lookup. The system invokes *IFSCP* only if the *WFSCP* cannot recognise the input string as well-formed. When the *IFSCP* identifies a local error to be substitution of a word, the spelling correction algorithm is invoked and provided with syntactic information inferred by *IFSCP*, to correct the word which is believed to be misspelt. This strategy has the advantage that the recovery process cannot affect the performance of a parser on well-formed sentences in terms of speed or complexity of parsing strategies.

This is an advantage in terms of processing efficiency: in terms of human processing, it corresponds to proposing that human processing has efficiency as a goal, and that consequently special methods that are not normally used are brought to bear on ill-formed sentences. Weischedel and Sondheimer would appear to be positing that error-repair methods are initiated exactly when the error is first detected - our approach corresponds to saying that it is reasonable to defer correction until more is known. This could be the end of the current phrase, or the current sentence, or perhaps just until a couple of further words have been seen. In the *WFSCP/IFSCP* system, *WFSCP* processing of the current sentence is completed before error correction is begun, but this is largely a matter of algorithmic convenience, and the system could be adapted so that it would try correction after the end of the current noun phrase, if an error was detected while a noun phrase was being processed (for example, because an unknown word was found).

We also employ a ranking system to suggest the best repair among many alternatives. There are two ranking strategies: syntactic level ranking and lexical level ranking. The syntactic rank is a penalty score derived from the importance of the repaired constituent in the local tree (e.g. head constituents are more important than modifiers) and the type of error correction (e.g. substitution < addition = deletion). The lexical rank depends on the distance between two letters on a keyboard. For example, *held* would be considered a more plausible replacement than *hold* for the non-word *hwld*, because *e* is closer to *w* on a standard keyboard than is *o*. Among humans, this type of correction strategy would of course only be available to those who were aware of keyboard layout.

Vosse (1992) attempted to correct an ill-formed sentence at the morpho-syntactic level. If there was a misspelt word, then his spelling corrector suggested the best correction. With this best correction his syntactic parser continued. Our system, on the other hand, employs a strictly top-down approach to a misspelt word. After *WFSCP* has finished trying to parse an ill-formed sentence containing a misspelt word, *IFSCP* is invoked and provided with the phrase structure, and misspelt word (if found by *WFSCP*). If *IFSCP* suggests that the error type is substitution of an unknown word, or (more difficult) a known word, then the lexical category inferred by *IFSCP* is used to assist in the spelling correction.

In section 2, we shall describe the structure of the system and in section 3, we outline results obtained when parsing ill-formed sentences. In section 4, we present conclusions and directions for extension of this work, and comment on the cognitive significance of the concepts developed in sections 2 and 3.

2 Design of *IFSCP*

In this section, we describe both parser stages, but focus on the error-recovery stage. The system is based on chart parsing using a context-free grammar. The system comprises two parsers, a syntactic grammar with non-static constraints, and a lexicon and verb subcategorisations based on the format of the Oxford Advanced Learner's Dictionary.

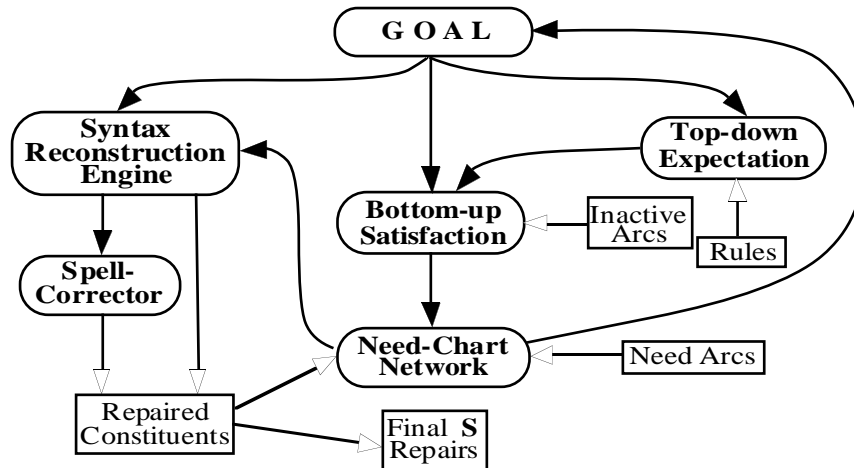


Figure 1. The Structure of IFSCP

The WFSCP follows a bottom-up parsing strategy and parses left-to-right. The IFSCP employs four major components (see Fig. 1):

- *top-down expectation*, which expands a goal to refine a local error, using syntactic rules based on a context-free grammar;
- *bottom-up satisfaction*, which refines a goal and builds a *need-chart network*, using inactive arcs from WFSCP;
- a *syntax reconstruction engine*, which repairs a detected errors and retraces the need-chart network reconstructing all the local trees; and
- a *spelling corrector*, which corrects spelling errors.

2.1 Top-down Expectation

In this section, we describe the type of structures built by IFSCP as it embarks on the top-down phase of its attempt to locate and classify an error, specifically the structures we call *goals*, and an associated measurement called a *minimum extension length score* (MELS).

If WFSCP classifies an input sentence as ill-formed, then IFSCP begins by hypothesising that the input sentence should be an S constituent: this S is the initial goal for IFSCP. By a *goal* we mean a local tree¹ which is believed to contain syntactic errors (the boxed constituents in Fig. 2). Such a goal is expanded by top-down expectation using context-free grammar rules to create subgoals - that is, subtrees, sub-subtrees, etc. of the original goal.

Goals contain the following information:

- a *label*: what phrasal constituent is needed for recovery;
- a *ps*: the penalty score for the goal;
- *parc*: which *need-arc* (defined in section 2.2) generates this particular goal; and
- *from* and *to*: the positions between which the goal constituent is needed.

Example: (goal: label=(adjp) from=1 to=2 ps=0 parc=narc5)

Goals are handled differently according to their characteristics (see Fig. 1): (a) a goal (such as S, NP1 NP2, ADJP in Fig. 2) may be made for the *top-down expectation* process; (b) a goal (such as ADJ in Fig. 2) may be handled by the *syntax reconstruction engine* to give a repaired local tree; (c) a goal such as (ADJP NOUN) in Fig. 2 may be processed by the *bottom-up satisfaction* process.

After a goal is generated, the relevant rules - those whose LHSs are the same as the goal's label and which satisfy the rule invocation condition (Min & Wilson, 1994) - are invoked to expand the goal. This is termed *top-down expectation*, and the expanded goal refines the error.

¹ We use the term *local tree* to refer to a subtree of a syntax tree corresponding to a single context-free rule.

When retrieving rules using the rule invocation condition, the *MELS* (Minimum Extension Length Score) between a goal and a rule is considered. The *MELS* is defined as follows:

$$\text{MELS}(G,R) = \text{WL}(G) - \text{MEL}(R)$$

where $\text{WL}(G)$ is the number of words between the two positions of the goal G (the *from* and *to*), and $\text{MEL}(R)$ is the Minimum Extension Length of the rule R . The MEL is the minimum number of lexical constituents from which the phrasal constituent can be constructed using the particular rule (e.g. the MEL of NP with $\text{NP} \rightarrow \text{PRON}$ is 1 (np (pron "I"))). For example, the MELS of an NP goal G from 0 to 3 with the rule $R: \text{NP} \rightarrow \text{DET NOUN}$, is 1, since $\text{WL}(G) = 3$ and $\text{MEL}(R) = 2$.

If $\text{MELS}(G,R)$ is greater than -2 , then the rule R is invoked for expansion of the goal G .

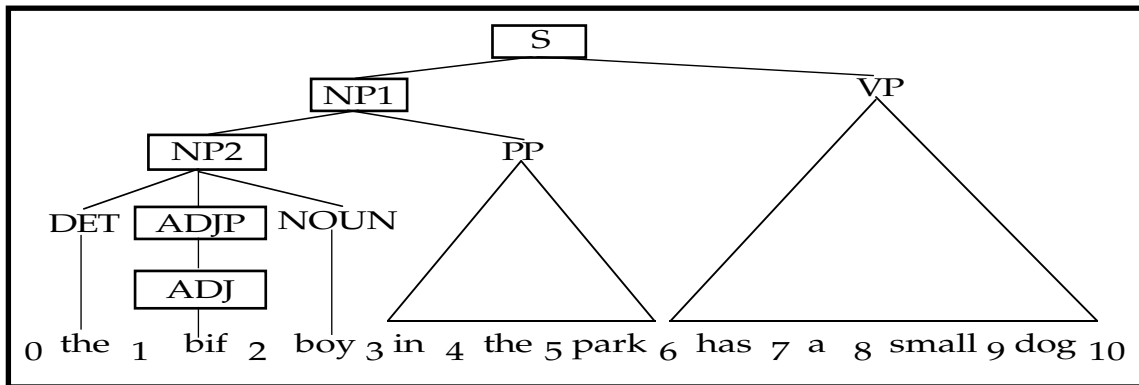


Figure 2. Goal production in a syntax tree.

The boxed constituents are those to be produced by the error correction process.

2.2 Bottom-up Satisfaction

Bottom-up satisfaction comprises two distinct steps: looking up inactive arcs, and creating the need-chart network. Our bottom-up satisfaction method uses a bidirectional algorithm (Stock, Falcone & Insinamo, 1988). After a goal has been expanded by top-down expectation, if an inactive arc is found that satisfies the leftmost or rightmost constituent of the expanded goal and any subcategorisation associated with the invoked rule (if the inactive arc is a verb), a need-arc is made using all the information from the expanded goal and the inactive arc that was found. The need-arc indicates that a constituent of a specified type, at a specified position, is required in order to complete a local tree. Here is an example of an NP need-arc showing a missing adjectival phrase (adjp):

```
(need-arc  nodename=narc5          label=np          ruletype=np4
          inactives=(det1 noun3)    neednode=(adjp)
          from=1 to=2              ps=0              parc=narc1)
```

As the example illustrates, need-arcs include the following information:

- which constituents are already found (*inactives*);
- which constituents are needed for recovery (*neednode*, this becomes a goal);
- the two positions between which there is a problem (*from & to*);
- what constituent should be repaired (*label*);
- its rule type (*ruletype*);
- the penalty score (*ps*) of the need-arc;
- what is the parent data structure of this need-arc (*parc*).

A need-arc is produced by one of three possible transformations: (a) a goal with known inactive arcs may advance an existing need-arc (see goal type (c) of section 2.1); (b) an expanded goal with known inactive arcs and invoked rules may make a new need-arc (see goal type (a) of section 2.1); and (c) if a need-arc satisfies the recovery condition for a local tree (see section

2.3), the syntax reconstruction engine is invoked to repair the local tree using the need-arc's information.

By these three transformations, need-arcs are made and linked together in a network to give hierarchical relations between the need-arcs (e.g. *parc* slot in a need-arc), containing localised error information linking the hypothesized S constituent to the bottommost level of the tree. This network is called a *need-chart network*; it links all need-arcs and the local trees recovered so far. The need-chart network allows direct access to needed data during the reconstruction of a repaired sentence. This network is accessed unidirectionally (from leaf to top node).

2.3 The Retraceable Need-Chart Network

The syntax reconstruction engine (SRE) allows recovery from local errors and produces repaired syntactic structures with a *deviance note* which explains what was done to effect the repair. It performs syntactic structure recovery by two distinct methods.

If a goal satisfies the *correction condition for goals*, namely that the category of the goal is a lexical category and the penalty score is either -1 (the addition case) or 0 (the substitution case), then it is processed by the SRE to produce repaired structures of the appropriate type. We term this method *correction from a goal node*.

The other method is termed *correction from a need-arc*. If a need-arc satisfies the *correction condition for need-arcs*, namely that all constituents of the need-arc are known but there is an extra word and its penalty score is 1 (the deletion case), the need-arc is repaired by deleting the extra word (e.g. need-arc = (np (det "the") (noun "boy"))) with *bif* 2 to 3 - cf. Fig. 2 - in this case another repair is also possible).

The SRE provides repaired constituents and allows recovery from specific errors in the bottommost local trees. The repairs can then be propagated or retraced up the tree via the need-chart network to complete the top-level (sentence) recovery.

2.4 Spelling Correction

If the SRE classifies a syntactic error as replacement of a word, then a spelling correction algorithm is invoked to correct the misspelt word. The algorithm is based on dictionary lookup and Damerau's research (1964), which indicates that most typographical errors are caused by one of single letter replacement, deletion, addition, or transposition of two adjacent letters. Our system implements this well-known principle with the restriction that the first letter is assumed correct but the first two letters can be transposed (so the system can handle *big* \rightarrow *ibg* but not *big* \rightarrow *hig*). For example, if the misspelt word is *bif*, then possible corrections are *biff*, *bid*, *big*, *bib*, *bit*, *bin*, and *biz*, given the coverage of our lexicon.

With syntactic information, our correction of the word *bif* can be more precise than this. For example, if the SRE has already found that *bif* must be an adjective, then (again given the coverage of our lexicon) the only possible correction is *big*.

All alternative corrections are ranked by their penalty scores. If a spelling correction is made, and the corrected word is in the lexicon, then the lexical information (e.g. number, form, and person) for the corrected word permits the generation of new inactive arcs as part of the repair. In the case of a substitution for an unknown word, if no correction is found by the spelling-corrector, then that syntactic repair is discarded. However, with substitution of a known word, failure to find a spelling correction does not cause the repair to be discarded. For example, *goes* cannot be spell-corrected to *go* in **the boys goes to school*, but we would not want to discard the attempt to repair *goes*. The word's lexical information is used to filter out implausible repairs at the syntactic level while retracing the need-chart network. At this stage, the non-static constraints such as subject-verb agreement and premodifier-noun agreement are applied to repaired constituents. Sometimes the constraints must be relaxed; e.g. if a detected problem is not corrected by spelling correction, then the repaired constituent simply consists of a suggested lexical category without any further information.

2.5 Ranking Schemes for Corrections

The system employs heuristics to select the best repair at both lexical and syntactic level. The system's basic strategy is to retain all alternatives for an error at a given level until other information is available to filter out implausible repairs. Both lexical and syntactic information are used. Thus syntactic information can be used to filter out implausible spelling corrections.

For example, if the unknown word *boi* is found, then the actual output of the spell-corrector is the list (*boo* 1.0) (*boil* 1.41) (*bop* 2.0) (*boy* 2.0) (*bon* 2.83) (*bog* 3.16) (*bob* 3.61) (*bow* 6.0) (*box* 6.32) (*boa* 7.07). The numbers indicate the keyboard distance between the new letter and the original letter in a replacement correction - for example, with *boi* → *boo* the replacement, *o*, is adjacent to the replaced letter, *i*, so the distance is 1.0. The list of ranked corrections can also be affected by grammatical knowledge - for example, if it were known that *boi* must be a verb, then with this extra information, the spell-corrector would return a shorter list: (*boo* 1.0) (*boil* 1.41) (*bob* 3.61) (*bow* 6.0) (*box* 6.32). These numbers can be viewed as *penalty scores* for the various alternative corrections.

(1) Penalty Scores at the lexical level

More specifically, we used a Pythagorean metric for keyboard distance based on the Qwerty keyboard layout. The keyboard is coordinatized - for example, if *q/Q* is arbitrarily chosen as having the coordinates (row, column) = (1,0), then those of *w*, *a*, and *n* would be (1,1), (2,0) and (3,5) respectively. The keyboard distance between *q* and *Q* is 0. The keyboard distance between *w* and *n* is $\|(3,5)-(1,1)\| = \sqrt{(2^2+4^2)} \approx 4.47$. This penalty system at the lexical level suits typographical error correction reasonably well, but a different one would be more appropriate for scanning errors, which might transform a word like *expect* to *e%pect*. The keyboard distance between % and *x* is 4.24. A metric based on letter similarity would be more appropriate here (and not difficult to implement). If the misspelling is caused by insertion or deletion of a letter, then the minimum of the letter distances between the inserted or deleted letter and the two adjacent letters is computed (e.g. in *boiy* vs *boy*, the letter distance between *o* and *i* is 1, and between *i* and *y* the distance is 2, so the minimum is 1). In the case of transposition errors, the letter distance between the two transposed letters is computed (e.g. the letter distance between *h* and *t* (*wiht* vs *with*) is 1.41).

(2) Penalty Scores at the syntactic level

The importance of a lexical category in the local tree is considered when ranking repairs at the syntactic level. If the system suggested more than one repair for a local tree, then a repair to a head daughter category has less penalty score than that of nonhead daughter categories. For each proposed repair, a penalty score is accumulated as processing proceeds from leaf to root until the final goal constituent (S) is reached. Then the penalty scores for the various alternative repairs are compared. In measuring penalty scores at the syntactic level, constituents are classified three ways: (i) if the repaired constituent is the head of a local tree, then the penalty score is 0.1. (ii) if the repaired constituent is a recursive constituent of a local tree and the constituent includes a head constituent (e.g. the NP in the RHS of the rule NP5 → NP PP), then the score is 0.3. (iii) if the repaired constituent is a nonhead constituent, then the score is 0.5. This penalty score is accumulated while retracing the need-chart network. For example, in figure 2, the SRE classifies the misspelt word *bif* as a substitution error with lexical category ADJ. A new repaired lexical constituent, ADJ3, gets penalty score 0.5 as this is the penalty for replacement of an unknown word. If the word *big* with spelling penalty score 1.0 is suggested by the spelling correction, then the penalty score of the repaired constituent ADJ3 (lhs=adj rhs="big" from=1 to=2 ps=1.5) becomes the sum of both penalty scores, 1.5 (= 0.5 (the error-type penalty score) + 1.0 (the penalty score for the spelling correction)). In figure 2, the penalty score of the repaired constituent ADJP is 1.6 (= 1.5 + 0.1 (the penalty score for the repaired head daughter ADJ3 in the local tree ADJP)). The penalty score for the repaired constituent NP2 is 2.1 (= 1.6 + 0.5 (the penalty score for a repaired nonhead daughter)). The penalty score of NP1 is 2.4 (= 2.1 + 0.3 (the penalty score for repair of a recursive constituent which includes a head daughter)). Finally, the penalty score of S is 2.9 (= 2.4 + 0.5 (the penalty score for repair of a nonhead daughter)).

(3) Penalty scores by error types

In addition to the two penalty schemes, we used some heuristics in the error-type part of our ranking system. If a word is not in the lexicon, then it is "recognised" as unknown. In the case of an unknown word, there are two possibilities at the syntactic level: either a misspelled word or an extra (junk) word. In the case of a known word error, there are three possibilities at the syntactic level; (i) the original word was misspelt, but the result is another word; (ii) an extra word (e.g. a repeated word); and (iii) a missing word. We give less penalty to substitution errors because most typographical or grammatical errors correspond to substitution errors. This is confirmed by Damerau's study (1964). The penalty scores we chose for repaired errors of these types are as follows:

- unknown word
 - replacement: 0.5
 - deletion: 1.0
- known word
 - replacement: 0.5
 - addition: 1.0
 - deletion: 1.0

The final selection of the best repair is based on penalty scores from the three ranking schemes: the three penalty scores are simply added. The system produces each alternative with its associated penalty score and the repair with the lowest score is chosen as the best repair. If two or more repairs tie for first place in this ranking, we have no further selection strategy at present.

3 Experimental Results

We conducted experiments to test the ability of our system to detect and correct syntactic/spelling errors. The test sentences used were ill-formed sentences with one of five types of error: substitution of an known/unknown word, addition of an known/unknown word, and deletion of a word, and four lengths (3, 5, 7, and 11) for each error type. Errors occur at all positions in the ill-formed sentences. Thus, in the case of a deletion error in a sentence of length 11, we have 11 test sentences, one for each possible deletion. In the case of errors involving only known words, the inserted or substituted words are randomly chosen. The basic sample sentences are "I have dogs", "The boy takes a book", "He wants to see the beautiful girl", and "He believes that the big boy enjoys seeing the small baby". Results² are shown in Table 1 and 2.

Error type **	No. of tesse sent.	fatal errors	first repair correct	second repair correct	later repair correct	no repair found	all repairs wrong
SUW	26	0	16 (62%)	4 (15%)	6	0	0
SKW	26	3	7 (27%)	3 (11%)	7	1	5
AUW	30	0	30 (100%)	0	0	0	0
AKW	30	4	12 (40%)	3 (10%)	6	1	4
DEL	26	8	7 (27%)	4 (15%)	5	1	1

Table 1. Test Results Classified by Error Type

** This refers to the transformation which was performed on an original (well-formed) sentence to provide a test sentence. SUW = Substitution of an Unknown Word, SKW = Substitution of Known Word, AUK = Addition of an Unknown Word, AKW = Addition of Known Word, DEL = Deletion of a word.

The system grammar has 38 context-free rules with 20 verb subcategorization rules and there are about 31,000 entries in the lexicon. The lexicon includes a wide variety of inflected and derived words. The average number of repairs found per sentence is 2.1 for errors caused by

² We used a Macintosh IIsi with 9MB RAM and Mac Common Lisp 2.0 for the experiments.

an unknown word and 4.1 for errors caused by a known word. In 15 of 138 (11%) sentences, the system does not recognise a syntactic error. In some cases, this is inevitable: for example, if the word *big* is deleted from the string *a big boy*, then the error cannot be detected. Similarly, if *big* is added to give *a big big boy*, it is impossible to recognise the error. We term these fatal errors: they occur when the mutated sentence is still well-formed syntactically.

Let us consider particular error types. When words were deleted (DEL), or replaced by a known word (SKW), in only 27% of the test sentences was the first repair suggested the correct one. The system gave the best result when an unknown word is added (AUW). In that case, for all the sentences, the first suggestion was deletion of the unknown word, and on average the system found only 1.5 repairs per testing sentence. The order of accuracy across ill-formed sentences was: addition of an unknown word > substitution of an unknown word > addition of a known word > deletion of a word > substitution of a known word (c.f. Table 2). This result differs markedly, and mostly for the better, from earlier results based on a context-free grammar (Min & Wilson, 1994) which did not apply spelling correction, a ranking strategy, non-static constraints, and verb subcategorization.

sentence length	test sentences	fatal errors	first repair correct	second repair correct	later repair correct	no repair found	all repairs wrong
3	17	0	13 (76%)	0	2	1	1
5	27	1	13 (48%)	5 (18%)	6	1	1
7	37	5	16 (43%)	5 (14%)	10	0	1
11	57	9	30 (53%)	4 (7%)	6	1	7

Table 2. Test Results Classified by Sentence Length

4 Conclusions

4.1 Performance Summary

The system is very robust in repairing a misspelt word which is caused by either the substitution or the addition of an unknown word: 81% of 56 testing sentences were corrected by using the first recommended repair and 89% by using either the first or second repair. Even though the system with context-free grammar and non-static constraints does not handle 3 cases, which have no repairs, the non-static constraints are quite good at filtering out many alternatives which are not plausible at the syntactic level.

When correcting spelling errors, it is very useful to integrate information about the lexical category of the mis-spelled word when this is deducible: this syntactic information reduces the number of implausible corrections at the syntactic level.

The heuristics we employed were successful in repairing errors involving substitution of an unknown word. Penalties based on the significance of a repair in the local tree appeared to work well. However, errors introduced by known words are not all detected and corrected successfully at the syntactic level.

4.2 Significance in Relation to Human Performance

We don't know exactly how humans repair errors: one-stage processing vs two-stage processing and error tolerance vs error correction. The two-stage error recovery approach does not affect the parsing of a well-formed sentence. The recovery system is invoked only when it is needed, and it proceeds to correct the error rather than to tolerate it.

However, the one-stage/two-stage distinction is not essential from a theoretical point of view. It would be possible, with a lot more effort, to implement a system which, for example, took smaller units of language, such as noun phrases, as top-level objects for repair (when necessary) - in such a case, the one-stage/two-stage distinction would largely disappear. An exception to this would arise with phrases whose first word was an unknown word - but then presumably humans, too, in such a case, would postpone diagnosis and repair until they had read/heard a few more words, in order to set the problem in context.

It seems obvious from introspection that humans do not consciously consider penalty scores, for example. However, it is plausible that they are aware at a subsymbolic level that certain types of error are more likely than others, in a way that is implemented explicitly in our system by the penalty score mechanism.

From a cognitive viewpoint, the basic point in section 2.5 is that humans are most likely to make errors in typing by striking keys adjacent to the correct ones or transposing or omitting keystrokes, producing bizarre-looking typos which people nevertheless manage to decipher. In the case of transposition errors, in particular, we feel that the most cognitively plausible errors are those where adjacent fingers on the same hand strike the key in the wrong order, or where fingers (most likely the same finger) on opposite hands strike the keys in the wrong order. Since adjacent fingers on the same hand will be striking keys close together on the keyboard, the Pythagorean keyboard metric will favour corrections produced in this mode. In the next section, we foreshadow improvements designed to take into account the "opposite hands" case. Similarly, confusions between the appearance of imperfectly printed or written characters (emulated by optical character recognition when it confuses I with 1, or S with 8) produce problems for people which our system seems able to resolve, in principle. It is, of course, difficult to see how to test whether humans are using the same similarity-metric strategies as we describe.

Because the human system integrates semantic and pragmatic constraints which we have not yet attempted to emulate, there are as yet plenty of examples of ill-formed sentences which humans handle but our system does not: we plan to move in this direction, as described in section 4.3.

4.3 Future Directions

If a semantically ill-formed sentence satisfies the system's grammatical and non-static constraints (as does **she becomes a car*), it is parsed as a well-formed sentence. In the case of IFSCP, the recovery covers both spelling and syntactic structure, but recovered versions of ill-formed sentences may be meaningless. We are trying to solve some of these problems, by enhancing the system to handle surface and deep case, using semantic selectional restrictions.

In spelling correction, there are a couple of directions that we plan to follow up. The first, mentioned in section 4.2, involves modifying the keyboard metric for transposition errors so that keys struck by the corresponding fingers on opposite hands are regarded as "close together." This would help solve the *teh* ↔ *the* problem: at present *teh* is likely to be corrected to *ten*, (a substitution error), because *h* and *e* are not very close together on the keyboard. This solution is also likely to be syntactically plausible: *ten houses* is likely to be acceptable wherever *the houses* would fit in.

We also plan to implement an alternative structural-similarity-based distance between characters. Thus, our system might not rank highly the natural corrections to errors likely to arise when using OCR technology or during manual or automated transcription of handwriting: % → x, \$ → s, I → l, O → 0. At the syntactic level, our system has a repair problem because of default left-to-right parsing.

When the error is repaired by a deletion of a word, and two adjacent words have the same lexical category, as in *a bonny boy boo* (*boo* is the extra word), the left word is always deleted. This problem arises algorithmically from the left-to-right processing, but could be addressed by a semantic level of processing - since (in terms of the example) a boy could well be bonny but a boo is not likely to be so. For example, a word-bigram approach might resolve this problem.

When the error occurs by deletion of a word (e.g. *I _ dogs*), the default left-to-right parsing cannot recognise a possible error between the words *I* and *dogs* and add a verb. This is because, in the bottom-up satisfaction process, instead of seeing *dogs* as a plural noun by right-to-left parsing, the system treats *dogs* as a verb (in third singular present form) by left-to-right parsing. The system finds problems (no object for a transitive verb, and a subject-verb agreement error), but they're the wrong problems. To cope with this problem, the bottom-up satisfaction needs more complex control schemes with static constraints for goals.

It is also possible that system performance could be improved by adjusting the parameters of the penalty score system.

References

- [1] Cooper, W., Tye-Murray, N., & Nelson, L. (1987). Detection of Missing Words in Spoken Text. *Journal of Psycholinguistic Research*, **16**(3), 233-240.
- [2] Damerau, F. (1964). A Technique for Computer Detection and Correction of Spelling Errors. *Communications of ACM*, **7**(3), 171-176.
- [3] Damerau, F. & Mays, E. (1989). An Examination of Undetected Typing Errors. *Information Processing & Management*, **25**(6), 659-664.
- [4] Fass, D. & Wilks Y. (1983). Preference Semantics, Ill-formedness, and Metaphor. *Amer. J. Comp. Linguistics*, **9**(3-4), 178-187.
- [5] Granger, R. (1983). The NOMAD System: Expectation-based Detection and Correction of Errors during Understanding of Syntactically and Semantically Ill-formed Text. *Amer. J. Comp. Linguistics*, **9**(3-4), 188-196.
- [6] Hayes, P. & Mouradian, G. (1981). Flexible Parsing. *Amer. J. Comp. Linguistics*, **7**(4), 232-242.
- [7] Mellish, C. (1989). Some Chart-based Techniques for Parsing Ill-formed Input. *ACL Proceedings, 27th Annual Meeting*, 102-109.
- [8] Min, K. & Wilson, W. (1994). Chart Parser for Ill-formed Input Sentences. *Language Teaching and Research*, **23**, 141-154, Language Research Center, Chonnam National University: Kwangju.
- [9] Shanon, B. (1973). Interpretation of Ungrammatical Sentences. *Journal of Verbal Learning and Verbal Behavior*, **12**, 389-400.
- [10] Stock, O., Falcone, R., & Insinamo, P. (1988). Island Parsing and Bidirectional Charts. *COLING-88*, 636-641.
- [11] Stock, O., Falcone, R., & Insinamo, P. (1989). Bidirectional Charts: A Potential Technique for Parsing. *Spoken Natural Language Sentences*, **3**, 219-237.
- [12] Vosse T. (1992). Detecting and Correcting Morpho-syntactic Errors in Real Texts. *ACL Proceedings, Third Conference on Applied Natural Language Processing*, 111-118.
- [13] Weischedel, R. & Sondheimer, N. (1983). Meta-rules as a Basis for Processing Ill-formed Input. *Amer. J. Comp. Linguistics*, **9**(3-4), 161-177.