Error Recovery in Natural Language Parsing with a Level-Synchronous Approach

Yi-Chung Lin* and Keh-Yih Su†

* Advanced Technology Center, Computer & Communication Research Lab.

Industrial Technology Research Institute, Chutung, Taiwan 310, R.O.C.

lyc@atc.ccl.itri.org.tw

† Behavior Design Corporation, 2F, No. 5, Industrial East Road IV

Science-Based Industrial Park, Hsinchu, Taiwan 300, R.O.C.

kysu@bdc.com.tw

Abstract

A level-synchronous probabilistic scoring function, which takes advantage of the wide-scope contextual information in the same phrase-level, is proposed in this paper to detect and recover the errors in ill-formed inputs. Traditionally, partial parsing is used in dealing with ill-formed inputs without fixing the errors. However, such an approach only provides coarse and limited information, which prevents the sentence from being processed further (such as translation). To fix the errors in ill-formed inputs, a recovery mechanism using the probabilistic scoring function is proposed in this paper. Experimental results show that 35% of the ill-formed inputs can be recovered to well-formed parses. The recall of constituent brackets is also significantly improved from 68.49% to 76.60%, while the precision of brackets is slightly improved from 79.49% to 80.69%.

1. Introduction

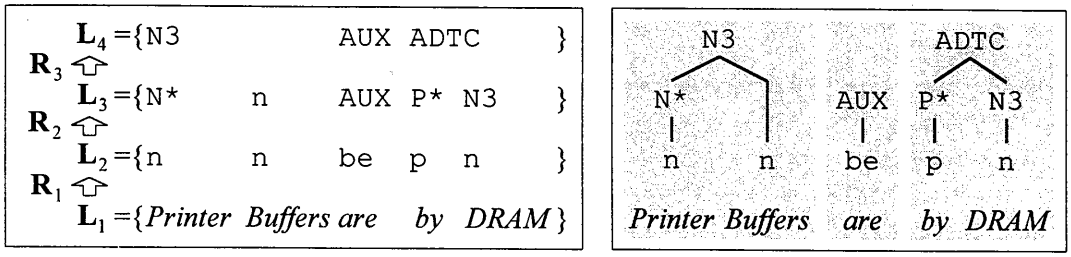
In real world applications, ill-formed inputs are inevitable in a natural language processing system. It is infeasible to limit users to speak or write with only a limited vocabulary or a predefined grammar. In many systems, ill-formed inputs are handled by a partial parsing mechanism. In these systems, an ill-formed input is first partitioned (or parsed) into recognizable pieces of phrases (i.e., partial parses). Then, by consulting the forest (Tomita, 1987) of the partial parses, the system takes some application-specific actions to deal with the ill-formed input. However, many errors in the inputs are not isolated from their neighbors; they may disguise their neighbors not to be recognized by the system. As a result, only coarse

and limited information is provided by partially parsing an ill-formed input if its errors are not fixed.

To recover the errors, some systems had tried to fix the errors during or after parsing. In 1981, Kwasny and Sondheimer (1981) proposed to parse ill-formed inputs with an Augmented Transition Network (ATN) parser. In their approach, the types of errors are carefully identified and the corresponding transition arcs, called relaxed arcs, are manually created in the network to recover those errors. These relaxed arcs are blocked in normal cases. Once all the grammatical paths fail, these relaxed arcs are attempted. Weischedel and Sondheimer (1983) also used a similar approach. They used meta-rules to associate certain ill-formed inputs with particular well-formed structures by modifying the violated grammar rules. In 1989, Mellish proposed to find the full parse by running a modified top-down parser over the partial parses generated by a bottom-up chart parser. The modified top-down parser attempts to find a full parse tree by considering one word error. On the other hand, in 1990, Abney (1990; 1991) proposed to parse natural language by segmenting the parts-of-speech into chunks and then assembling the chunks into a complete parse tree. In his work, the chunks were repaired and assembled by predefined heuristic rules. Recently, Lee et al. (1995) generalized the least-error recognition algorithm (Lyon, 1974) to find the full parses of minimum error with a small grammar of only 192 grammar rules. Since exhaustively finding the full parses with minimum error is very time-consuming, they used heuristic rules and heuristic scores to cut down the search space.

All the above mentioned approaches involve some ad hoc heuristic rules to fix the errors or restrict the search space. Those heuristic rules are usually system-specific and hard to be reused by other systems. Besides, although those approaches may work well in small tasks on specific domains, they lack extensibility and are hard to be scaled-up. Therefore, a generalized approach, independent of any particular system and domain, is highly demanded.

In this paper, we propose an error recovery mechanism using a generalized probabilistic scoring function to identify and recover the errors. Since the errors could occur at any places in the inputs, exhaustively searching all possibilities is infeasible. Thus, a two-stage strategy is proposed to limit the search space. In the first stage, the most possible forest of partial parses is tried to fit into the S-productions, whose left-hand side symbols are the "S" symbol (i.e., the start symbol). If the forest cannot be well fitted by applying one or two modification actions, the part-of-speech errors are considered in the second stage. Experimental results



(a) (b)

Figure 1. (a) Parsing by building phrase-levels. (b) The corresponding partial parses.

show that 35% of the ill-formed inputs are recovered to their correct well-formed full parses. The recall of constituent brackets is increased from 68.49% to 76.60%, while the precision of brackets is slightly improved from 79.49% to 80.69%.

2. Robust Parsing by Building Phrase-levels

In natural language processing, contextual information is helpful for resolving ambiguity problems. One obvious example is the trigram part-of-speech tagger (Church, 1989). In order to incorporate wide-scope contextual information into the parsing process, we (Lin, 1995; Lin and Su, 1997) proposed to parse a sentence in a level-synchronous manner, in which parsing a sentence is treated as building a set of *phrase-levels*.

Figure 1 is an example of the building process. Initially, the first (i.e., the lowest) phrase-level, $\mathbf{L}_1$, consists of the input words. Then, according to the dictionary, $\mathbf{L}_1$ is built-up to the second phrase-level, $\mathbf{L}_2$, which consists of parts-of-speech. After then, phrase-levels are built-up according to the grammar rules. This process repeats until no more rules can be applied. If the input is well-formed, we can get the "S" symbol in the topmost phrase-level; otherwise, the topmost phrase-level will contain the roots of the partial parses derived from the input. As a result, a parse tree, either a full parse or a forest of partial parses, can be represented as $\mathbf{T} = \{\mathbf{L}_N, \mathbf{R}_{N-1}, \mathbf{L}_{N-1}, \cdots, \mathbf{R}_1, \mathbf{L}_1\}$, where N is the number of phrase-levels, $\mathbf{L}_i$ indicates the i-th phrase-level, and $\mathbf{R}_i$ denotes the set of actions (i.e., reduce actions) used to build $\mathbf{L}_{i+1}$ from $\mathbf{L}_i$.

We also proposed the following scoring function to evaluate the likelihood of a parse tree:

$$S_{\mathbf{T}} = \prod_{A_j \in \mathbf{L}_N} P(A_j \mid A_{j-2}, A_{j-1}) \times \prod_{i=1}^{N-1} \prod_{\rho_j = \langle r; l \rangle \in \mathbf{R}_i} P(r \mid A_{l-1}^{l+1}), \tag{1}$$

where A_j is the j-th symbol in $\mathbf{L}_N$, r is the production rule applied by the j-th action ρ_j in $\mathbf{R}_i, A_{l-1}^{l+1}$ is the short-hand for "A_{l-1}, A_l, A_{l+1}" in $\mathbf{L}_{i+1}$. Intuitively, the first term $\prod_{A_j \in \mathbf{L}_N} P(A_j \mid A_{j-2}, A_{j-1})$ accounts for the prior probability of the topmost phrase-level; the second term $\prod_{i=1}^{N-1} \prod_{\rho_j=<r;l> \in \mathbf{R}_i} P(r \mid A_{l-1}^{l+1})$ ascribes to the likelihood of applying the grammar rules.

The proposed level-synchronous approach is tested on 200 ill-formed English sentences collected from an English-to-Chinese machine translation system. Compared with the approach of using the stochastic context-free grammar and the heuristics of preferring the longest phrase (Mellish, 1989; Seneff, 1992), the level-synchronous approach improves the precision and recall of brackets from 69.37% to 79.49% and from 78.73% to 81.39%, respectively (Lin, 1995; Lin and Su, 1997).

3. Error Recovery with Modification Actions

The above level-synchronous approach does not fix any errors. However, many errors of ill-formed inputs are not isolated from their neighbors. They may affect their neighbors and prevent them from being recognized by the parser. In order to reduce the effects of errors, a recovery mechanism should be incorporated. For this purpose, the building process is further generalized to fix the errors of ill-formed inputs as follows.

To fix the errors in a phrase-level, the modification actions of insertion, deletion and substitution are incorporated into the parsing process. These actions are realized by three modification productions: $X \rightarrow \varepsilon$ (inserting a symbol X), $\varepsilon \rightarrow X$ (deleting a symbol X) and $Y \rightarrow X$ (replacing the symbol X with the symbol Y). A modification action consists of a rule argument and two position arguments, like $\tilde{\rho} = \langle \tilde{r}; u, v \rangle$, where $\tilde{r}$ stands for the applied modification production rule, u and v indicate that this modification action is applied between the u-th and the v-th symbols in the modified phrase-level.

For example, the input "*Printer Buffers are by DRAM*" missed a verb after the auxiliary "*are*". To correct such an error, a verb should be inserted so that $\mathbf{L}_2$={n n be p n} could be modified to $\tilde{\mathbf{L}}_2$={n n be **v** p n}. In this case, the corresponding modification action set $\tilde{\mathbf{R}}_2$ will consist of one modification action $\tilde{\rho} = \langle \tilde{r} = \mathrm{v} \rightarrow \varepsilon; u = 3, v = 5 \rangle$. In that

modification action, the rule argument $\tilde{r} = \mathrm{v} \rightarrow \varepsilon$ indicates to insert a verb; the position arguments u=3 and v=5 indicate the inserted verb is placed between the third and the fifth symbols of the modified phrase-level $\tilde{\mathbf{L}}_2$.

Incorporated with modification actions, the parsing process can be considered as applying modification actions and normal actions in turn. Therefore, a modified parse tree $\tilde{\mathbf{T}}$ of N phrase-levels can be represented as $\tilde{\mathbf{T}} = \{\mathbf{L}_N, \mathbf{R}_{N-1}, \tilde{\mathbf{L}}_{N-1}, \tilde{\mathbf{R}}_{N-1}, \mathbf{L}_{N-1}, \cdots, \mathbf{L}_2, \mathbf{R}_1, \tilde{\mathbf{L}}_1, \tilde{\mathbf{R}}_1, \mathbf{L}_1\}$, where $\mathbf{L}_i$ is the i-th phrase-level; $\tilde{\mathbf{R}}_i$ is the set of modification actions to modify $\mathbf{L}_i$; $\tilde{\mathbf{L}}_i$ is the result of applying $\tilde{\mathbf{R}}_i$ to modify $\mathbf{L}_i$; $\mathbf{R}_i$ is the set of normal actions applied to build $\mathbf{L}_{i+1}$ from $\tilde{\mathbf{L}}_i$. After some derivations like those deriving Equation (1) (Lin and Su, 1997), the score of a modified parse tree is defined as

$$S_{\tilde{\mathbf{T}}} = \prod_{A_j \in \mathbf{L}_N} P(A_j \mid A_{j-2}, A_{j-1}) \times \prod_{i=1}^{N-1} \prod_{\rho_j = <r;t> \in \mathbf{R}_i} P(r \mid A_{t-1}^{t+1}) \times \prod_{\tilde{\mathbf{R}}_i = \phi} P(\tilde{\mathbf{R}}_i) \times \prod_{\tilde{\mathbf{R}}_i \neq \phi} \prod_{\tilde{\rho}_j = <\tilde{r};u,v> \in \mathbf{R}_i} P(\tilde{r} \mid \tilde{A}_u^v), \tag{2}$$

where the notation $\tilde{\mathbf{R}}_i = \phi$ indicates that $\tilde{\mathbf{R}}_i$ is an empty set (i.e., no modifications are applied to modify $\mathbf{L}_i$); $\tilde{\mathbf{R}}_i \neq \phi$ denotes that it is not an empty set, $\tilde{A}_u^v$ is the short-hand for symbols from the u-th to the v-th in $\tilde{\mathbf{L}}_i$. The first two product terms, which are related to the normal productions, are the same as those in Equation (1). The third product term $\prod_{\tilde{\mathbf{R}}_i = \phi} P(\tilde{\mathbf{R}}_i)$ is related to the phrase-levels which need not be modified. The last product term $\prod_{\tilde{\mathbf{R}}_i \neq \phi} \prod_{\tilde{\rho}_j = <\tilde{r};u,v> \in \mathbf{R}_i} P(\tilde{r} \mid \tilde{A}_u^v)$ accounts for the modification actions. Currently, the probabilities in Equation (2) are estimated from an annotated corpus by using Good-Turing estimation method (Good, 1953).

4. Two-stage Strategy to Find Potential Modification Actions

Theoretically, any modification actions can be applied to modify any phrase-levels of an ill-formed input. However, since the number of possible modifications is very large, it is infeasible to blindly try every one. Therefore, a two-stage strategy is proposed to find the

potential modifications. In the first stage, the forest of partial parses is first fitted into the S-productions, whose left-hand-side symbols are the "S" symbol (i.e., the start symbol). If the forest of partial parses cannot be well fitted into those S-productions, they are passed to the second stage to recover the part-of-speech errors. The details of these two stages are described in the following sections.

4.1 Fitting the partial parses

The errors of ill-formed inputs are two kinds: the isolated errors and the clingy errors. The isolated errors are those that do not hinder the parser from correctly parsing other phrases. Take the sentence "*Printer buffers are made by DRAM*" as an example. If the noun phrase "*Printer buffers*" is missed, the other words still can be parsed into a verb phrase. Therefore, such an error is an isolated error. On the contrary, missing the word "*made*" will hinder a parser from parsing the other three words "*are by DRAM*" into a verb phrase. Thus, it is a clingy error (clings to other words) in this example.

Isolated errors could be recovered by fitting the partial parses (Jensen, Miller, and Ravin, 1983). In the past, the fitting procedure is usually guided by heuristic rules, such as preferring some head phrases and preferring the widest phrase. Since acquiring those heuristic rules is expensive and maintaining the consistency of a large number of rules is difficult, the heuristic approach is hard to be scaled up. Besides, the heuristic rules are usually system-specific and hard to be reused by other systems. Therefore, we attempt to recover the isolated errors by fitting the partial parses according to probabilistic scores.

The forest of partial parses is fitted into the S-productions because most of the partial parses are constituents of S-productions. At most two modification actions are allowed to fit the forest of partial parses to the right-hand sides of the S-productions. Different modification actions can fit the forest into different S-productions and construct different full parse trees. These full trees are then ranked by the scoring function $S_{\tilde{T}}$ in Equation (2). If the forest of partial parses cannot be fitted into a full tree with one or two modification actions, it is assumed that the errors of this ill-formed input are not isolated. Then, the partial parses are passed to the second stage to fix the clingy errors.

4.2 Recovering errors of parts-of-speech

It is noticed that many clingy errors come from the second phrase-level (i.e., the phrase-

level of parts-of-speech). Therefore, in the second stage, attempts are made to recover the errors originated from the second phrase-level. Since enormous modifications can be applied to modify parts-of-speech, it is infeasible to try all of them. To be practical, currently, only the modifications with one insertion, deletion or substitution are permitted. Furthermore, since our training set of ill-formed inputs is rather limited, it cannot offer reliable statistical information to find the potential actions to modify the parts-of-speech. Therefore, the statistical information is acquired from well-formed training data via three scoring functions as described below.

Using the trigram formulation, the likelihood of a part-of-speech sequence, $c_1^n = c_1, c_2, \cdots, c_n$, can be approximated to be $\prod_{j=1}^{n} P(c_j \mid c_{j-2}, c_{j-1})$, where c_j denotes the j-th part-of-speech. Therefore, the score for inserting a part-of-speech "x" before the i-th part-of-speech is defined as:

$$S_{\text{INS}}(i, x; c_1^n) \equiv P(x \mid c_{i-2}, c_{i-1}) \times P(c_i \mid c_{i-1}, x) \times P(c_{i+1} \mid x, c_i) \times \prod_{\substack{j=1 \\ j \neq i, i+1}}^{n} P(c_j \mid c_{j-2}, c_{j-1})$$

With this scoring function, we can find the most probable modifications of one insertion action. Currently, only the top 5 insertion actions are applied to modify parts-of-speech. In the same way, the scores for deleting the i-th part-of-speech and substituting the i-th part-of-speech with "x" are respectively defined as:

$$S_{\text{DEL}}(i; c_1^n) \equiv P(c_{i+1} \mid c_{i-2}, c_{i-1}) \times P(c_{i+2} \mid c_{i-1}, c_{i+1}) \times \prod_{\substack{j=1 \\ j \neq i, i+1, i+2}}^{n} P(c_j \mid c_{j-2}, c_{j-1})$$

$$S_{\text{SUB}}(i, x; c_1^n) \equiv P(x \mid c_{i-2}, c_{i-1}) \times P(c_{i+1} \mid c_{i-1}, x) \times P(c_{i+2} \mid x, c_{i+1}) \times \prod_{\substack{j=1 \\ j \neq i, i+1, i+2}}^{n} P(c_j \mid c_{j-2}, c_{j-1})$$

Again, only the top 5 deletion actions and the top 5 substitution actions are applied to modify the parts-of-speech. These 15 modified phrase-levels of parts-of-speech are then parsed by building the modified phrase-levels and finally the full parse trees are ranked by the scoring function $S_{\tilde{T}}$ defined in Equation (2).

5. Experimental Results and Discussions

In our experiments, 8,727 well-formed sentences collected from computer manuals and their correct parse trees are used as the training data to estimate the parameters corresponding to the normal production actions. The average length of these sentences is about 13 words. A context-free grammar of 29 terminals, 140 nonterminals and 1,013 productions is used to parse input sentences. To estimate the parameters corresponding to the modification actions in Equation (2), another training set of 300 ill-formed sentences is collected and carefully parsed to full parse trees annotated with the required modification actions. Besides, the 200 ill-formed sentences in the testing set are also parsed to full parse trees with correct modification actions so that they can be used to test the performance of the proposed error recovery mechanism.

Table 1 lists the experimental results of parsing the 200 ill-formed testing sentences. The first row (PLB) corresponds to the performance of parsing without error recovery. The second row (ER1) gives the results of error recovery up to the first stage (i.e., fitting the partial parses). The last row (ER2) shows the results of error recovery up to the second stage (i.e., both fitting the partial parses and recovering errors of parts-of-speech).

The second row in Table 1 shows that, by fitting the partial parses into the S-productions, 25% of the ill-formed inputs can be correctly parsed and fitted to full parse trees. The last column indicates that 43% of the ill-formed inputs can be parsed to full parse trees by fitting their partial parses with one or two modification actions. In other words, 18% (resulted from subtracting 25% from 43%) of the ill-formed inputs are parsed to incorrect full parse trees.

The last row of Table 1 shows that, using the two-stage error recovery mechanism, 35% of the ill-formed inputs can be correctly parsed to the full parse trees. That is, 10% (resulted from subtracting 25% from 35%) of the ill-formed sentences are correctly parsed by recovering the errors of parts-of-speech. An ill-formed sentence correctly recovered in the

	Bracket and its label		Parse tree	
	Precision	Recall	Accuracy	Fitting rate
PLB	79.49%	68.49%	0.0%	0.0%
ER1	80.02%	70.59%	25.0%	43.0%
ER2	80.69%	76.60%	35.0%	76.0%

Table 1. Performances of parsing without and with error recovery

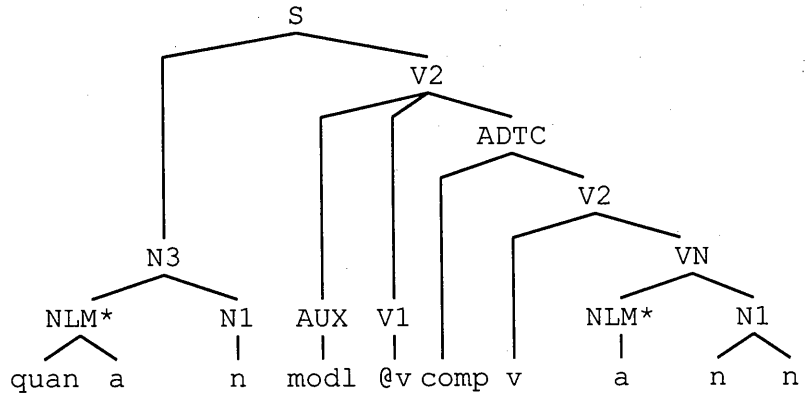


Figure 2. An ill-formed sentence correctly recovered in the second stage. "@V" indicates to replace the part-of-speech of the word "sufflce" with "v".

second stage is shown in Figure 2. In this example, the word "sufflce", which is a typo of the word "suffice", is regarded as an unknown word and, thus, is considered as a noun by default. After the second stage, 76% of the ill-formed sentences can be fitted into a well-formed syntactic structure.

By carefully inspecting the results, we find that, at the first stage, the improvement on the accuracy rate of parse trees is significant but the improvement on the bracket recall rate is little. The bracket recall rate is significantly improved in the second stage. This phenomenon is due to the fact that the errors recovered at the first stage are isolated errors. These errors do not hinder the parser from correctly parsing other words. On the contrary, the errors recovered at the second stage are not isolated. They seriously affect the partial parses of other words. Therefore, recovering such errors can significantly improve the bracket recall rate.

At both the first stage and the second stage, some ill-formed sentences are parsed to the incorrect full parse trees. Therefore, the numbers of incorrect brackets are increased at both stages. While computing the precision rate for brackets, these increased incorrect brackets compensate the increased correct brackets, which come from correctly parsing the ill-formed sentences. As a result, there are almost no improvement on the bracket precision rate at both the first stage and the second stage.

6. Error Analysis and Future Works

Although 35% of the ill-formed sentences can be recovered and correctly parsed to full parse trees, there are still many errors unresolved. By inspecting the remaining 65% unrecovered

sentences, three different types of errors are identified.

About 35% of the overall errors are caused by syntactic ambiguity. Most of these syntactic errors come from the problem of prepositional phrase attachment. To resolve such type of errors, purely syntactic information is not enough; higher level language knowledge, such as semantic knowledge, should be incorporated.

On the other hand, about 29% of the overall errors are due to incorrectly applying modification actions. Most of these errors occur at the second stage, where modification actions are applied to modify the parts-of-speech. About half of these errors are introduced because the correct modification actions are not included in those top 15 modification actions. To attack this problem, the discrimination power of the scoring function, which is used to select the potential modification actions, should be enhanced. Another half of the errors come from selecting the full parse tree derived from the undesired modification action. For example, the correct modification action for the ill-formed sentence "*An may appear in the display*" is to insert a noun after the word "*An*". However, the output tree is derived from the undesired modification action which replaces the part-of-speech of the word "*An*" with pronoun. This is because the incorrectly modified full tree is better than the correctly parsed full tree from the syntactic point of view. Therefore, such errors can be regarded as coming from "syntactic ambiguity" and the semantic knowledge should be incorporated to resolve them.

The last part of the unrecovered errors is caused by multiple errors in an ill-formed input. The portion of this part is 36%. For example, the sentence "*The tutorial explains how and why to use the tool*" is not covered by our grammar. Our grammar just covers the noun clause of only one question word (i.e., "*why*") followed by a infinitive (i.e., "*to use the tool*"). The desired modifications for this sentence is to delete the parts-of-speech of the words "how and". To recover such errors, multiple modification actions should be allowed to modify the parts-of-speech in the second stage. Since the number of different ways to modify a phrase-level of parts-of-speech with multiple modification actions is very large, a fast search method is necessary to find the likely combinations of modification actions. Besides, applying multiple modification actions will cause the numbers of parts-of-speech of the modified phrase-levels to be very different. However, a full parse tree with fewer parts-of-speech is usually more likely to have a higher score than a full tree with more parts-of-speech. Therefore, the normalization issue must be considered to fairly score the full parse trees with different numbers of parts-of-speech.

In summary, further improvements should be made in the following directions. First, semantic knowledge should be incorporated to resolve the errors coming from syntactic ambiguity and some of the errors caused by incorrectly applied modification actions. Second, the discrimination power and speed of the scoring function for searching potential modification actions should be enhanced. Third, the normalization issue should be considered to deal with the ill-formed sentences of multiple errors.

7. Conclusion

By incorporating wide-scope contextual information, the level-synchronous parsing mechanism (Lin and Su, 1997) showed its superiority in efficiency and accuracy for parsing ill-formed inputs. However, the proposed mechanism does not try to recover the errors. It only provides a forest of partial parses for an ill-formed input. In this paper, we generalize the level-synchronous parsing mechanism such that the errors can be fixed and a full parse can be provided. Besides, a two-stage strategy is also used to efficiently find the most probable modification actions to recover the errors in an ill-formed input. The experimental results show that the enhanced parser can correctly recover the errors in 35% ill-formed inputs. The recall of brackets is significantly improved from 68.49% to 76.60%, while the precision of brackets is slightly improved from 79.49% to 80.69%.

Acknowledgements

This paper is a partial result of both the Project 3P11200 conducted by ITRI under sponsorship of the Ministry of Economic Affairs, R.O.C. and the Project NSC-84-2221-E-007-013 sponsored by the National Science Council, R.O.C.

References

Abney, S. P., "Rapid Incremental Parsing with Repair," in *Proc. of the 6th New OED Conference: Electronic Text Research*, 1990, pp. 1–9.

Abney, S. P., "Parsing by Chunks," in *Principle-Based Parsing: Computation and Psycholinguistics*, Robert C. Berwick, Steven P. Abney, and Carol Tenny (Eds.), Kluwer Academic Publishers,1991, pp. 257–278.

Church, K. W., "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text," in *Proc. of ICASSP*, 1989, pp. 695–698.

Good, I. J., "The Population Frequencies of Species and the Estimation of Population Parameters," Biometrika, 40, 1953, pp. 237–264.

Jensen, K., G. E. Heidorn L. A. Miller, and Y. Ravin., "Parse Fitting and Prose Fixing: Getting a Hold on Ill-formedness," *American Journal of Computational Linguistics*, 9(3–4), 1983, pp. 147–160.

Kwasny, S. C. and N. K. Sondheimer, "Relaxation Techniques for Parsing Grammatically Ill-Formed Input in Natural Language Understanding Systems," *American Journal of Computational Linguistics*, 7(2), 1981, pp. 99–108.

Lee, K. J., C. J. Kweon, J. Seo, and G. C. Kim., "A Robust Parser Based on Syntactic Information," in *Proc. of the 7th Conference of European Chapter of the Association for Computational Linguistics*, 1995, pp. 223–228.

Lin, Y.-C., "A Level Synchronous Approach to Ill-Formed Sentence Parsing and Error Recovery," Ph.D. Thesis, Department of Electrical Engineering, National Tsing-Hua University, ROC, 1995.

Lin, Y.-C. and K.-Y. Su., "A Level Synchronous Approach to Ill-Formed Sentence Parsing," in *Proc. of the 10th R.O.C. Computational Linguistics International Conference*, 1997, pp. 89-108.

Lyon, G., "Syntax-Directed Least-Errors Analysis for Context-Free Languages," *Communications of the ACM*, 17(1), 1974, pp. 3–14.

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

Seneff, S., "Robust Parsing for Spoken Language System," in *Proc. of ICASSP*, 1992, pp. 189–192.

Tomita, M., "An Efficient Augmented-Context-Free Parsing Algorithm," *Computational Linguistics*, 13(1-2), 1987, pp. 31-46.

Weischedel, R. M. and N. K. Sondheimer, "Meta-Rules as a Basis for Processing Ill-Formed Input," *American Journal of Computational Linguistics*, 9(3–4), 1983, pp. 161–177.