A Three-level Revision Model for Improving Japanese Bad-styled Expressions

Yoshihiko HAYASHI
NTT Network Information Systems Laboratories
1-2356, Take, Yokosuka, Kanagawa, 238-03, Japan
E-mail : hayashi@nttnlz.ntt.jp

Abstract

This paper proposes a three-level revision model for improving badly-styled Japanese expressions, especially in the field of technical communication. The model is a mixture of the regeneration-based model and the rewriting-based model. The first level divides long sentences, while the second level improves several badly-styled expressions with iterative partial rewriting operations. The last level performs regeneration, in which word ordering and punctuation to reduce the reading ambiguity are currently involved. Experimental results show that our model is effective in realizing practical revision support systems.

1 Introduction

It is well known that "revision is a large part of the writing process "[1]. To provide computational aids for revision, several style-checkers and revision support systems have been developed[2],[3]. However, few systems have the capability of providing alternatives for the expression determined to be badly styled[4]. In addition, these systems simply show the alternative expressions, the user must rewrite the original sentence while referring to the suggested expressions.

We have developed a prototype of our sentence-level Japanese revision support system called REVISE-S[5]. In the system, the user can improve his/her sentences by simply selecting the most appropriate alternative from the candidates that the system generates.

This paper proposes a three-level revision model for improving badly-styled Japanese expressions. We focus on the field of technical communication. An architecture of the revision support system based on the model is presented. Experimental results are shown that prove the effectiveness of the prototype system and the validity of the proposed model.

2 Computational Aids to Revision

2.1 Targets of the Computational Aids

Mishima[6] has summarized the key conditions for efficient technical communication via texts as follows:

(1) The reader must be able to *easily* understand the text (Easy-understanding).

(2) The readers must be able to *correctly* understand the text (Correct-understanding).

(3) The contents of the text must meet the reader's purposes.

From this viewpoint, the task of revision is text rewriting or regenerating to make the original text satisfy these conditions. The last condition is too hard to support computationally; however, the first two conditions are promising because we can apply natural language processing technologies. Therefore, we concentrate on the first two conditions in designing the computer-assisted revision system.

2.2 Revising as Regeneration or Rewriting

Is there just one computational revision model which is suitable for use? Two models illustrated in Fig.1 are the two extremes and will give the basis for constructing a practical model: one is the *Regeneration-based model* (a) and the other is the *Rewriting-based model* (b).

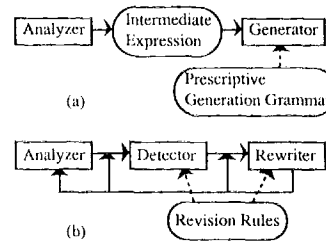


Fig.1 Two Basic Revision Models.

The regeneration-based model is a strong model; namely, if all of its components are perfectly constructed, the output text will be understandable and contain no badly-styled expressions. However, if some component is incomplete, there is a possibility of some input text being fatally flawed. Therefore, for this model to be practical, as a minimum, the following two major problems must be overcome:

(1) The Analyzer must correctly capture the intermediate representation of the input text at a certain processing level.

(2) The Generator must be equipped with the complete set of prescriptive generation grammar.

The first problem is serious, especially in revision support systems, and hard to be overcome. This is because input text may contain badly-styled expressions that prevent correct computational analyses. Moreover, a solution to the second problem is also problematic, because no perfect sets of prescriptive generation grammar have been developed to date. Furthermore, even if such a set could be developed, single-pass generation has been pointed out to have many drawbacks in producing optimal texts[1]. In spite of these problems, the regeneration-based model is crucial for offsetting the weaknesses of the rewriting-based model.

In the rewriting-based model, on the other hand, the original text is iteratively rewritten to improve each badly-styled expression that has been detected; only detected expressions are revised. This means that the rewriting-based model is a weak but practical model. Even if the set of revision rules is incomplete, the revision process will not destroy the original text entirely; the worst case is that the revision will be insufficient. Moreover, if the set of revision rules successfully cover numerous badly-styled expressions, it is expected that the system can achieve good performance. Thus, if most of the considered style improvements can be handled with this model, we should combine it with the regeneration-based model.

3 Classification of Japanese Badly-styled Expressions

It is obvious from the previous discussion that the effectiveness of the rewriting-based model depends on how many style improvements can be described as individual revision rules. Thus we must investigate what patterns of expressions should be considered to be badly-styled, especially in the technical communications field, and determine how many of them can be improved by revision rules.

Table.1 Classification of Typical Japanese Bad-styled Expressions.

Item (Example)	Correct-Understanding (C) / Easy-Understanding (E)	Linguistic Level (Scope)	General (G)/ Technical Writing (T)	Improvable with Revision Rule
Too Long Complex Sentences	C = E	Sentence	G	N
Unclear Inter-clause Connective Expressions コマンドを投入し、結果を転送する。	C < E	Two Clauses	T	Y
Operation Directing Expressions with Reverse Step システムを立ちあげる前に電源を投入する。	C < E	Two Clauses	T	
Partial Prohibitory Expressions コピーする時以外は、触らないでください。	C < E	Two Clauses	T	
Improper Voices (trans./intrans, passive/active) プログラムを動作できない。	C < E	Clauses	G	
Double Negatives 操作できないことはない。	C < E	Two Clauses	T	
Ambiguous Negatives with Comparing Expression XシステムのようにＣＰＵパワーを消費しない。	C < E	Clauses	T	
Ambiguous Negatives with Quantifier 全部のファイルが印刷できないときは、－－－	C < E	Clauses	T	
Conditional Expressions with Negated Antecedent and Negated Consequence 電源を投入しないと動作しない。	C < E	Two Clauses	T	
Violated Concord Expressions (Adjective and Predicate) 決して計算には成功します。	C = E	Clauses	G	
Violated Concord Expressions (Subject and Predicate) システムの特徴は、高速で計算します。	C < E	Sentence	T	
Light Verb Expressions 分散システムの迅速な開発が行えます。	C < E	Clauses	T	
Ambiguous Modification Structures	C > E	Sentences	G	N

To investigate these issues, we have classified typical sentence-level Japanese badly-styled expressions. The classification was mainly used examples from several books on technical writing[6],[7] as well as general writing[8]. Textual data from published manuals on computer systems was also investigated. The result is briefly outlined in Table.1. The viewpoints for classification are:

(1) Whether the item affects easy-understanding or correct-understanding?

(2) In which linguistic structure does the item occur?

(3) Is the item general or peculiar to technical writing?

(4) Can the item be improved with an individual revision rule?

The investigation showed that items peculiar to technical writing mainly affect easy-understanding, while general items principally affect correct-understanding. In addition, most of the items peculiar to technical writing can be improved by the application of discrete revision rules. Fig.2 exemplifies a revision rule for a typical badly-styled expression peculiar to technical writing; the expression directs the user's actions, but the actions are described in reverse order. We can identify most badly-styled expressions peculiar to technical writing by referring to particular partial syntactic structure patterns. As shown in Fig.2, such patterns allow bad-styles to be detected and rewritten. Therefore, it is valid to adopt the rewriting-based model as the center component of our model.

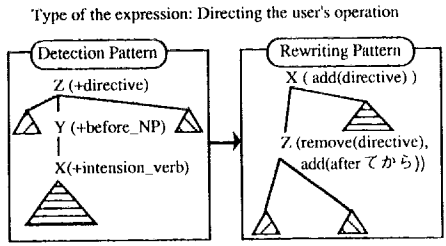


Fig.2 A Partial Rewriting Operation as Structural Conversion with Lexical Operations.

4 The Three-level Revision Model and the Prototype System

4.1 The Model

The previous section has shown that the rewriting-based model is applicable for most of the style improvements peculiar to technical writing. Table.1, however, shows that there are a couple of items which are poorly handled by the model. They are:

(a) excessively long complex sentences, and

(b) ambiguous modification structures.

These items cannot be detected and corrected by the particular revision rules, because they do not have unique syntactic patterns. These errors cannot be characterized by particular words and/or particular linguistic attributes such as, modality, tense, etc.. Thus these badly-styled expressions cannot be easily corrected with the particular structural conversion operations.

We are proposing a three-level revision model which combines the rewriting-based and regeneration-based models. The first level is for dividing excessively long complex sentences and is based on the regeneration-based model at the morphological level. The second level is for improving several badly-styled expressions and is based on the rewriting-based model. The third level is for syntactic/semantic level regeneration, in which word ordering and punctuating to reduce the number of structural ambiguities are involved.

Our model is a three-level sequential model. Here, the order of the components has the following computational significance:

(1) As shown in 5.1, excessively long complex sentences can be identified and divided with morphological level information[9].

(2) If long sentences are divided at the early stage of the total process, processing loads for the remaining operations are significantly reduced.

(3) The style improving process should precede the syntactic/semantic level regeneration process, because the regeneration process should start with a well-formed syntactic/semantic structure.

4.2 Issues in Improving Style

Most style improvements can be realized by sequential application of the revision rules. However, there are two major design issues. One is how to feedback the result of each rewriting operation to the initially produced analysis results. The other is the handling of structural ambiguity. That is, if the ambiguity is not eliminated, combinatorial explosion is inevitable in many aspects of the system. On the other hand, overall structural disambiguation is computationally expensive due to processes such as semantic analysis and context analysis. Moreover, uniform application of these processes violates one of the basic requirements of any writing aid; that is, it is unacceptable to incur high computational costs by processing good expressions that require no revision.

We have three approaches to deal with these issues:

(1) First, we detect all of the potential bad styles while accepting structural ambiguity. Each bad style is connected to an associated partial rewriting operation specified by its pattern. These operations are defined in a rule-base, so that the detection process is the activation of these rules.

(2) We then try to apply activated rules under an expectation-driven control strategy. That is, the system schedules the order of rule applications using a priority that reflects how important the rewriting operation is in improving the sentence. The scheduled application of a rule initiates the structural disambiguation of the applicable expression.

(3) During the revision process, internal data, such as that generated by morphological or syntactic analyses and by the bad-style detection process, varies as a result of the partial rewriting operations. To avoid duplicative

analysis and detection, we accurately know what has been revised, and ensure the consistency of the internal data with respect to the revision. This scheme solves the feedback problem mentioned before.

4.3 The Architecture of the Prototype System

Figure.3 shows the architecture of the prototype system REVISE-S based on the three-level revision model and the above design principles.

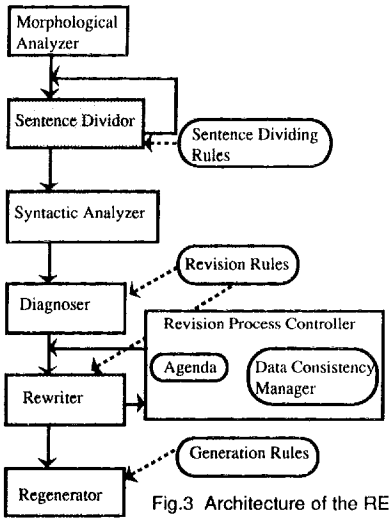


Fig.3 Architecture of the REVISE-S

The Morphological Analyzer divides the sentence string into word sequences. At this time, basic operational units (called 'Bunsetsu') are recognized. The sentence dividing algorithm in the Sentence Dividor utilizes the result of the morphological analysis, and is outlined in 5.1. The sentence dividing process is recursively invoked until each divided sentence satisfies some predefined condition that prevents further division.

Next, the Syntactic Analyzer finds all possible binary relations between modifier Bunsetsu and modified Bunsetsu. The result is represented in a network called a Kakari-Uke network which represents all possible syntactic structure intensionally.

The Diagnoser, which utilizes the detection counterpart of the revision rule, finds all possible badly-styled expressions. The result semi-fires the conversion counterpart of the associated revision rule and constructs the agenda which lists the semi-fired rule instances. The Revision Process Controller sequences the successive execution of partial rewriting operations, and the Data Consistency Manager maintains consistency between the current sentence string and the internal data during the dynamic rewriting process.

Finally, the regeneration process is invoked to generate a sentence with less reading ambiguity.

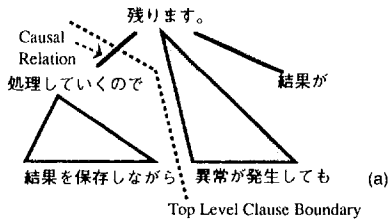
5 Generating Alternative Expressions

Each component in our revision model generates alternative expressions for the user. This section gives a brief outline of the generation of alternative expressions in each level component.

5.1 Dividing Long Sentences

Before dividing a long complex sentences, first the component must decide whether the sentence should be divided or not. If the sentence is so determined, then, the component must identify the division point. The top level clause boundary indicates the division point. Finally the divided sentences must be generated. These processes can be conducted with morphological level information; that is, they do not require full syntactic parsing or any semantic interpretation.

In the first step, the decision is made with a discriminate function that computes the weighted sum of the number of characters, the number of Bunsetsus and the number of predicates (verbs and adjectives), etc.. Weighting coefficients and the threshold value for decision were determined through experiments.



(b-1)
結果を保存しながら処理していきます。
The process advances while saving the result.
したがって、異常が発生しても結果が残ります。
Thus the result remains, even if error occurs.
(b-2)
異常が発生しても結果が残ります。
The result remains, even if error occurs.
これは、結果を保存しながら処理していくためです。
Because the process advances while saving the result.

Fig.4 An Example of the Sentence Division.

The second step roughly analyzes the intra-sentence connective structure[9] and produces a shallow level intermediate representation as illustrated in Fig.4(a). The key to this process is the inter-predicates dependency relation analysis which utilizes a set of dependency rules. These rules are based on the classification of predicate expressions (including modal, tense, aspectual suffixes) in terms of the strength in forming connective structures. One significant point in the process is that the connective structure must not be fully disambiguated, because the main purpose of the analysis is identification of the division point; namely, there are cases where the division point can be uniquely identified, nevertheless the connective structure is ambiguous.

The final step generates the divided sentences string by applying generation rules to the intermediate representation. Fig.4(b) gives the generated alternatives ((b-1),(b-2)) for the example in Fig.4(a). In the process, ordering of the divided sentences and choice of the conjunctive expression which provide cohesion between divided sentences are major considerations. In Fig.4(a), two top level clauses are connected with a causal relation. Thus associated conjunctive expressions (underlined in Fig.4(b)) are generated according to the alternatives in sentence ordering. To determine which is better, contexual processing is required; however the determination is currently left to the user's selection.

5.2 Rewriting through Partial Structural Conversions

Main stream of the algorithm in style improvement component is summarized in Fig.5. The rest of this subsection briefly introduces topics in each step (details are given in [5]).

```
Detect all Possible Bad-styled Expression ;
Construct the Agenda and
  the Revision Process Manager ;
while (T) do
   Select an unmarked rule instance with the highest
     priority from the agenda ;
   if there is no such rule instance then
     break ;
   Test its presupposition ;
   if the presupposition holds then {
     Apply the associated partial rewriting operation ;
     if the operation succeeds then
       Analyze the difference and
         maintain data consistency ; }
   Mark the instance as "done" ;
end while ;
```

Fig.5 Main Stream of the Algorithm in Style Improvement Component.

Detection of Badly-styled Expressions

The Diagnoser detects badly-styled expressions from the Kakari-Uke network which contains all detectable syntactic structures. The process is the semi-firing of the partial rewriting rules, because each detected badly-styled expression is associated with a rewriting rule specified by the type of the bad-style pattern. 'Semi-firing' means that some of the focused rules are deactivated later in response to on-demand structural disambiguation or partial rewriting. From the computational viewpoint, the detection process should be regarded as a sort of feature extraction process. This allows the diagnosis process to be realized as an interpretation of the data-flow network; namely, the terminal node finally activated indicate which associated badly-styled expression has been detected and node own data provides justification.

Constructing Agenda and Revision Process Manager

The rules semi-fired through the process described in the previous section are instantiated based on their justifications. The instances are then placed on the agenda. These justifications specify the partial syntactic structures concerned with the detection patterns. Therefore, these are presuppositions to the application of the associated rewriting operations.

A justification is represented as a conjunction of predicates for modification relations between two Bunsetsus (called the Kakari-Uke condition) and predicate on the Bunsetsu properties (called the Bunsetsu property condition). For instance, the conjunctive formula stated below is the justification of the detection pattern shown in Fig.2.

```
+intension_verb(X) ∧ +before_NP(Y) ∧
+directive(Z) ∧ modify(X,Y) ∧ modify(Y,Z)
```

The literal of the formula is called a primitive condition. Literals must be treated as a sort of assumption, because all of them have the possibility of becoming unsatisfied due to structural disambiguation and/or partial rewriting operations.

The Revision Process Manager for managing the presuppositions is constructed at the same time as the agenda. It holds a list of Bunsetsu property conditions and a list of the Kakari-Uke conditions. The data structure is suitable for managing all presuppositions systematically because rule instances that share the same primitive condition are immediately found through a data slot which is indexed by the primitive conditions, and contains pointers to the rule instances.

Expectation-Driven Control and On-demand Disambiguation

The priorities preassigned to the instances on the agenda sequence the successive application of partial rewriting operations within the revision process. That is, important rewriting operations are assigned high priority values, and are scheduled to for earlier application, even if their presuppositions are not confirmed prior to their application. To actually apply a scheduled rewriting operation, its presupposition is tested first. At this time, the Disambiguator which involves the application of heuristic disambiguation rules and/or user-interactions is invoked, and the minimum range of structural ambiguities is resolved in expectation of applying the scheduled rewriting operation.

Partial Rewriting Operation

If the presupposition is confirmed to be satisfied, the associated partial rewriting operation is applied. Before commencing any partial rewriting operation, a sub-network concerned with scope of the rewriting is first extracted from the Kakari-Uke network according to the given scope name such as 'simple sentence' and 'noun phrase', etc.. Second, the extracted sub-network is converted into a set of dependency trees, wherein each element is an explicitly represented possible syntactic structure. Third, the partial rewriting rule defined by the structural conversion with the lexical operations is applied to the trees. Alternative expressions are generated from rule application. A partial rewriting operation is completed by user selection or rejection of the generated expressions. The partial dependency trees giving the selected partial expression are then converted to the sub-network again, and restored in the Kakari-

Uke network. This process is iterated until the agenda has no more applicable rule instances.

Maintenance of Data Consistency with Constraint Propagation

Because the structural disambiguation and the partial rewriting operations affect internal data, the system must maintain the consistency of internal data whenever these operations are invoked. Brand-new information may be obtained as a result of the invoked operations, i.e., the acceptance/rejection of some modification or the change of some Bunsetsu structure. The new information can be considered as newly added constraints so that data consistency can be maintained by propagating these constraints to the dependent internal data.

For instance, if the Revision Process Manager is notified by the Difference Analyzer that a particular Bunsetsu no longer has a certain property according to a particular partial rewriting operation, the rule instances which share the condition are immediately deactivated. Another typical example of the constraint propagation is created by structural disambiguation. If some Kakari-Uke relation is confirmed by the Disambiguator, exclusive Kakari-Uke relations are rejected at this time. This causes the deactivation of the rule instances which have these rejected Kakari-Uke relations as their primitive conditions.

5.3 Word Ordering and Punctuating as Regeneration

Appropriate word ordering and punctuating help reduce the ambiguity in reading. Furthermore, it increases readability. In Japanese, however, word order is relatively free at the sentence constituent level and there are no strict grammatical restrictions on punctuation. Thus optimal word order and punctuation can not be decided only with syntactic information; reading and writing preferences must be considered.

Our regeneration algorithm takes the syntactic structure (dependency tree structure) as its input and regenerates a new syntactic structure with less reading ambiguities. The algorithm employs the following heuristics based on the preferences in word ordering[10]:

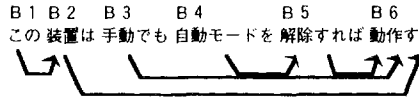
(1) Constituents which include the thematic marker (post position 'ha') are put at the head of the sentence, and punctuation marks are put after them.

(2) Punctuation marks are placed on clause boundaries.

(3) Heavier constituents (containing more Bunsetsus) are made to precede light constituents on the same syntactic level.

The algorithm first determines the constituent which includes the thematic marker. The constituent is positioned at the head of the sentence and a punctuation marker (Japanese comma) follows it. Next, the punctuating mark is added to the Bunsetsu which indicates the top level clause boundary. Then, at each syntactic level, constituents are sorted by their weight. Of course, the initially located constituents that include thematic markers are not moved by this constituent sorting operation. Finally, if the regenerated sentence string differs from the original, it is submitted for user confirmation.

Figure.6 gives an example. In this example, B2 is the Bunsetsu that contains the thematic marker and B5 indicates the top-level clause boundary. According to the regeneration algorithm, the segment (B1-B2) is placed at the head of the sentence and the Japanese comma is added. The segment (B4-B5) precedes segment (B3) because of its weight (two Bunsetsus) and another comma is added.



→ この装置は、自動モードを解除すれば、
手動でも動作する。

This device will work manually, if the automatic-mode has been canceled.

Fig.6 An Example of the Regeneration.

6 Evaluation

An evaluation experiment to show the effectiveness of the prototype system and the validity of the proposed revision model was made by using 113 sentences taken from published manuals and constructed examples. The points for evaluation were how much the the system contributes to easy-understanding and correct-understanding.

6.1 Readability

There is no established way to evaluate understandability of texts. In this paper, we treated understandability as roughly equivalent to readability, because readability is encompassed by understandability.

The readability measure used in the experiment was proposed by Tateishi,et.al[11] for Japanese texts. The method computes the readability with the following formula which utilizes surface level information. The term RS' indicates the readability; higher values indicate the text is more readable. The coefficients were determined through statistical analyses to normalize the mean value to 50 and the standard deviation to 10.

RS' = -0.12 x ls - 1.37 x la + 7.4 x lh - 23.18 x lc - 5.4 x lk -4.67 x cp + 115.79

These terms are, ls: length of the sentences, la:mean length of alphabetical characters run, lh: mean length of Hiragana characters run, lc: mean length of Kanji character runs, lk: mean length of Katakana character runs, cp: mean number of commas per sentence.

The system increased the RS' value by 42.5 to 49.0. This means that the readability was increased by the system. Sentence division and punctuation were the main contributors to this improvement.

6.2 Structural Ambiguity

It is also difficult to quantitatively estimate correct-understanding. In this paper, we estimate the level of correct-understanding from the structural ambiguity, because structurally ambiguous sentences/expressions

obviously degrade correct-understanding. However, measuring systematic[12] or reading ambiguity with algorithms is still a difficult problem. Thus we use computational ambiguity to approximate systematic ambiguity. The Japanese dependency structure analyzer developed by Shirai[13] was used for this purpose.

The original texts led to 18.4 analyses per sentence on average. After the texts were corrected by the prototype system, only 7.9 analyses were produced. This means that the system successfully reduced the amount of structural ambiguity. The major contributors to this improvement were sentence division and word ordering. Style improvements leading to drastic changes in the syntactic structure also contributed to this improvement.

Incidentally, after revision, only 4.9 possible syntactic structures remained per sentence on average within the internal data of the system. This is a fair bit less than the result from the reanalysis of the revised text. Thus where the revised text is processed further (for instance, translation, summarization), the use of the internal data will help to reduce the effect of disambiguations on the remaining processes.

6.3 Validity of the Model

The validity of the proposed revision model was not directly evaluated in the experiments. However, the validity of the component order is evident, because structural ambiguity is continuously reduced with each processing step. If the style improvement component preceded the sentence division component, the structural conversion processes to improve the badly-styled expressions would handle numerous fruitless syntactic structures and generate too many inappropriate alternative expressions. Moreover, if the syntactic/semantic regeneration component preceded the style improvement component, each structural conversion rule would be constructed as to preserve the word order and punctuation marks; this would affect the writability of the rules.

7 Concluding Remarks

This paper has proposed a three-level revision model for improving badly-styled Japanese expressions and introduced a prototype revision support system based on the model. Experimental results show that the system successfully improves the readability of texts and reduces the contained structural ambiguities. The three-level model effectively realizes a practical revision support system.

However, a remaining requirement from the real technical writing field is that expert knowledge from technical writers should be accumulated to cover a wider variety of badly-styled expressions. In addition, contextual information must be handled, both for providing contextually adequate alternative expressions and for improving contextually-poor expressions and rhetorical structures.

The proposed model and the prototype system as its embodiment, will give a powerful foundation to some other applications, including intellectual tutoring systems and pre-editing systems for machine-translation.

Acknowledgments

The author wishes to extend his gratitude to Gen-ichiro Kikui, who developed the rewriting rule application mechanism and to Eiji Takeishi, for his contributions in developing the sentence division module. Thanks are also due to the members of the Message Processing Systems Laboratory for their helpful discussions.

References

[1] Vaughan,M.M. and McDonald,D.D. (1988) A Model of Revision in Natural Language Generation System, *Proc. of the 26th Annual Meeting of the Association for Computational Linguistics*, 90-96.

[2] Thurmair,G. (1990) Parsing for Grammar and Style Checking, *Proc. of the 13th International Conference on Computational Linguistics*, 365-370.

[3] Richardson,S.D. and Barden-Harder, L.C. (1988) The Experience of Developing a Large-scale Natural Language Text Processing System: CRITIQUE, *Proc. of the 2nd Conference on Applied Natural Language Processing*, 195-202.

[4] Hakomori,S. et al. (1988) A Correction System for Japanese Text (in Japanese), *IPSJ SIG-NL*,65-7.

[5] Hayashi,Y. (1991) Improving Bad Japanese Writing Styles through Partial Rewriting Operations, *Proc.of the Natural Language Processing Pacific Rim Symposium*, 30-37.

[6] Mishima,H. (1990) *Technical Writing for Engineers and Students* (in Japanese), Kyouritsu-Shuppan, Tokyo.

[7] Technical Communication Associates (Eds.) (1988) *An Exciting Stylebook for Ducumentation* (in Japanese), NIKKEI-BP, Tokyo.

[8] Iwafuchi,E. (Eds.) (1988) *Bad-styles* the 3rd Edition (in Japanese), Nihon-Hyouronsya, Tokyo.

[9] Takeishi,E. and Hayashi,Y. (1990) A Method to Decide Division Points of Japanese Complex Sentences (in Japanese), *Proc. of the 4th Annual Conference of JSAI*, 9-6.

[10] Saeki,T. (1975) *Word Order in Modern Japanese* (in Japanese), Kasama-syoin, Tokyo.

[11] Tateishi,K. et al. (1988) Derivation of Readability Formula of Japanese Texts (in Japanese), *IPSJ SIG-DPHI*, 18-4.

[12] Hindle,D. and Rooth,M. (1991) Structural Ambiguity and Lexical Relations, *Proc. of the 29th Annual Meeting of the ACL*, 229-236.

[13] Shirai,S. (1987) Table-driven Japanese Phrase Dependency Analysis in Japanese-to-Englush Translation System ALT-J/E (in Japanese), *The 34th Annual Convension IPS Japan*, 5W-5.