Japanese language analysis for syntactic tree mining to extract characteristic contents

Yohsuke Sakao, Takahiro Ikeda, Kenji Satoh and Susumu Akamine
Media and Information Research Laboratories, NEC Corporation
1753, Shimonumabe, Nakahara-ku
Kawasaki, Kanagawa, Japan , 211-8666
{y-sakao@bu, t-ikeda@di, k-satoh@da, s-akamine@ak}.jp.nec.com

Abstract

Existing syntactic ordered tree mining methods for extracting characteristic contents from text sets have two problems: 1) subtrees which are semantically the same but are different ordered trees fail to be considered equivalent, and 2) raw extracted subtrees can be difficult to understand. In order to avoid these problems, we have developed a method of transforming all ordered trees so that the ordered trees having the same meaning are considered equivalent. We have also developed a method of constructing Japanese texts from extracted subtrees, and evaluated the effectiveness of our methods as applied to syntactic tree mining.

1 Introduction

Text mining technology is widely used as a framework for discovering knowledge from such large text data bases as questionnaire and contact-center data bases.

Text mining methods employing the syntactic structures of texts have recently been proposed. Some of them restrict parse trees to ordered trees in order to enumerate efficiently all subtrees. For example, Kudo et al. have proposed an enumerating algorithm which restricts parse trees to ordered trees (Kudo et al., 2004) in order to enumerate efficiently all subtrees through the use of a rightmost expansion (Asai et al., 2002). With respect to practical use, however, these text mining methods present the following two problems. (1) Under the above restriction, subtrees which are different ordered trees cannot be counted together, even when they in fact have the same meaning. This is a serious drawback. (2) Further, the meaning of raw extracted subtrees can be difficult to understand.

In this paper, we use a Japanese text mining method which handles dependency trees and lists subtrees through the use of a rightmost expansion (Asai et al., 2002). We hereafter call this text mining method "tree mining". A dependency tree is a tree whose nodes represent phrases and whose edges represent dependencies between phrases. For each node label, we use the base form of a representative word in the phrase corresponding to the node. In order to avoid the problems encountered in existing tree mining methods that use ordered trees, we first developed a method of transforming all trees so that the ordered trees having the same meaning would be the same. Further, we have developed a new method of constructing Japanese texts from extracted subtrees in order to obtain mining results that are easy to understand.

This paper is organized as follows. In Section 2, we explain in detail problems with existing tree mining systems that use ordered trees. In Section 3, we present our tree mining method. In Section 4, we describe the prototype tree mining system that we have developed. In Section 5, we report evaluation results for the prototype system. In Section 6, we conclude with a summary and a description of planned future work.

2 Problems with Tree Mining Methods that Use Ordered Trees

Existing tree mining methods which use ordered trees suffer from following two drawbacks.

2.1 Trees Which Should Be Counted Together Cannot Be If They Are Different Ordered Trees

The three trees shown in Figure 1 below illustrate the problem, which here may be considered specific to Japanese (i.e., the same problem would not necessarily occur in the treatment of other languages). Here, for the purposes of text mining, the phenomenon of expensive cars being bought by young people represents a single “meaning,” and while each of the trees expresses that meaning, their orderings are all different. Specifically, (a) *“Jakunensoh-Ga Kohkyuhsha-Wo Kohnyu.”* refers to young people’s buying of expensive cars, (b) *“Kohkyuhsha-Wo Jakunensoh-Ga Kohnyuh”* refers to the same thing, but the respective locations of the subject and object in the Japanese

sentence have been reversed, and (c) *"Jakunensoh-Ga Kohnyuh-Suru Kohkyuhsha"* refers to expensive cars bought by young people. As may be seen in the figure, in dependency trees, differences in the order of dependency are represented by differences in the ordering of sister nodes, while differences in the direction of dependency are represented by differences in edge directions.

2.2 Difficulty of Understanding the Meaning of Raw Extracted Trees

With mining systems that output characteristic subtrees, it is often difficult for users to understand the meaning of the subtrees, particularly when the subtrees have a large number of nodes.

3 Methods

In order to overcome the above drawbacks, we incorporate the following two methods into tree mining: (1) structural transformation, and (2) sentence construction from extracted characteristic subtrees.

3.1 Structural Transformation of Dependency Trees

After parsing, we conduct the following two structural transformations: (1) sister node reordering, and (2) root expansion.

3.1.1 Re-ordering of Sister Nodes

Here, we unify the ordering of sister nodes by re-ordering them all on the basis of a single rule; in this case, we put sister labels in alphabetical order. As may be seen in Figure 2, this results in the order of Figure 1's (a) and (b) becoming the same. In this way, we are able to normalize ordered trees with regard to the order of these sister nodes.

If the dependency tree contains sister nodes with the same label, we construct trees for all permutations of the sister nodes.

Note that dependency trees which have different meanings can become the same tree through this reordering. We evaluate the validity of this reordering in Section 5.1.

3.1.2 Root Expansion

Here, we construct a separate dependency tree for each node in the original tree, with that node forming the new tree's root. Such a "root expansion" is illustrated in Figure 3. This eliminates the problem created by differences in edge direction. We can enumerate all patterns of dependency directions thorough root expansion. We use root expansion because the direction of dependency cannot easily be normalized. Figure 4 shows an example of failed dependency direction normalization that was based on alphabetical order. In this example, using alphabetical order, we adopt node *A* as a root node of Tree 1, and we adopt node *B* as a root node of Tree 2. While, before

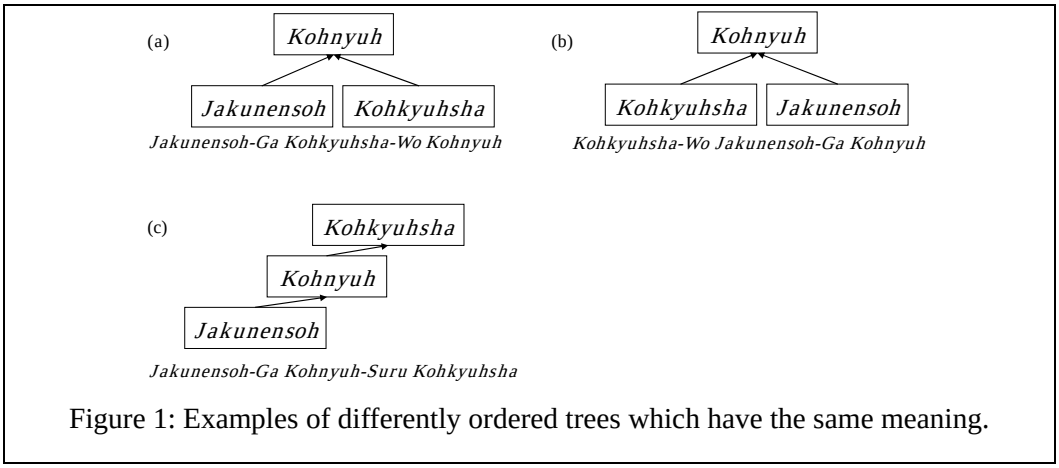


Figure 1: Examples of differently ordered trees which have the same meaning.

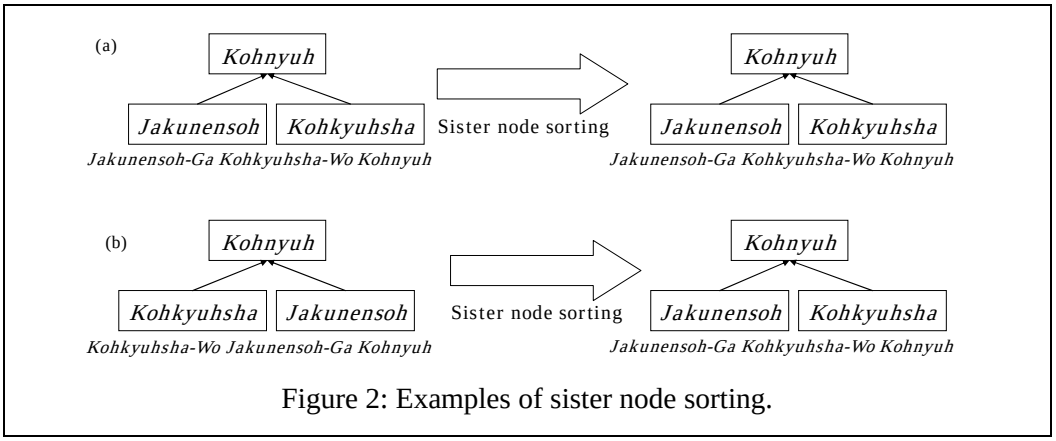


Figure 2: Examples of sister node sorting.

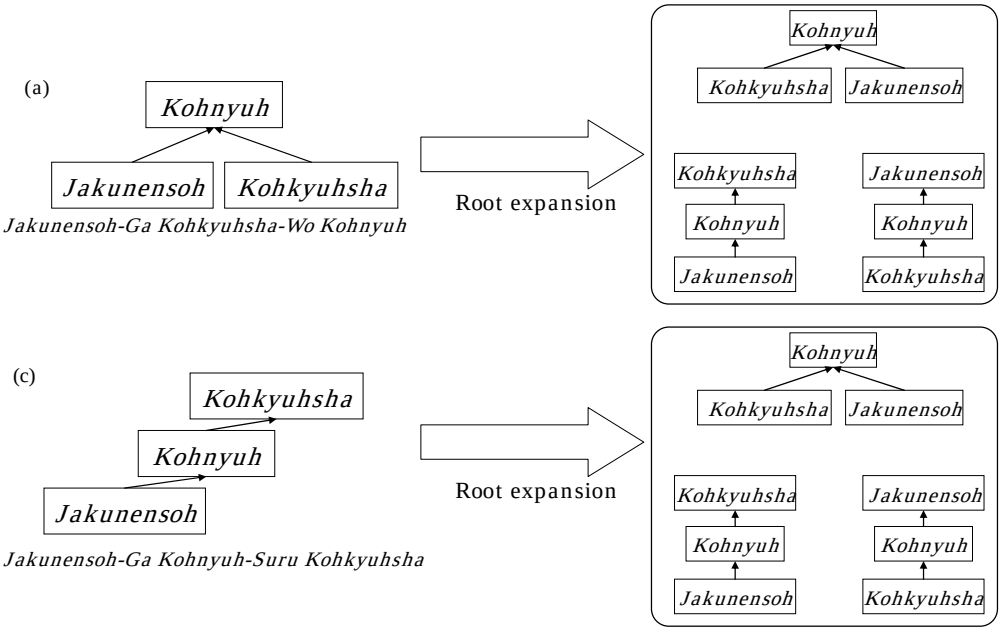


Figure 3: Examples of root expansion.

dependency direction normalization, Tree 1 contained Tree 2 as a subtree, after the normalization, Tree 1' did not contain Tree 2'.

Suppose N is the average number of nodes in dependency trees. Tree mining with root expansion will then be about N times slower than tree mining which restricts its dependency trees to ordered trees. It will still, however, be faster than tree mining which does not restrict its parse trees.

Dependency trees which have different meanings can become the same tree by means of this root expansion. We evaluate the validity of this root expansion in Section 5.1.

3.2 Sentence Construction from Extracted Characteristic Subtrees

After conducting the above structural transformations, we extract all characteristic subtrees and then generate candidate sentences for each characteristic subtree by arranging the original phrases (represented as subtree nodes) in the order found in the original text. Next, we calculate a score for each candidate sentence on the basis of a phrase bigram model, and the candidate with the highest score is selected for output as the sentence corresponding to the characteristic subtree in question.

Specifically, an output sentence S is selected from sentence candidates $\{W \mid w_1 \ldots w_n\}$ as follows:

$$S = \arg\max_W P(W) \approx \arg\max_W P(w_i \mid w_{i-1}),$$

where $w_1,\ldots,w_n$ are phrases in sentence candidate W, $P(W)$ is the probability that W will occur, $P(w_i|w_{i-1})$ is the bigram probability that phrase w_i will succeed phrase w_{i-1}.

Bigram probabilities are estimated from a large corpus and/or the input text sets of the tree mining system.

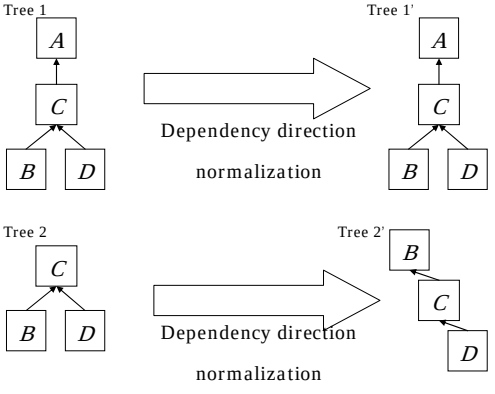


Figure 4: Example of failed dependency direction normalization.

4 Prototype System

Our newly developed prototype system executes tree mining in the following steps:

1) Parsing: It constructs dependency trees by parsing input text sets.

2) Structural Transformation of Dependency Trees: It applies the structural transformation introduced in Section 3.1 to the dependency trees constructed in the parsing.

3) Extraction of Characteristic Subtrees: It uses a rightmost expansion (Asai et al., 2002) to list up all subtrees in each dependency tree, and then calculates a score for each subtree on the basis of information gain (Morinaga et al. 2005). One or more subtrees are then selected as characteristic

subtrees. The selection may be based on ranking (e.g., best 3 scores, etc.) or on a pre-determined threshold value (e.g., scores exceeding 0.5, etc.).

4) Deleting Unneeded Characteristic Subtrees:

In order to avoid the output of characteristic subtrees whose meanings are essentially the same, the prototype deletes all but one of any subtrees which would be the same if they were considered as free trees rather than as ordered trees; it leaves only the subtree with the highest score. Further, it also deletes any subtrees that are actually already contained within other subtrees.

5) Sentence Construction from Extracted Characteristic Subtrees:

Finally, the prototype uses the method described in Section 3.2 to construct, from extracted subtrees, sentences to be output.

5 Evaluation

In our evaluation of the prototype system, we applied it to the following two corpuses.

Corpus 1: A business report data base containing 19,399 texts. Average text size is 60.4 characters (characters include both ideographs (kanji) and syllable-characters (kana)).

Corpus 2: A corporate contact-center data base containing 1,197 texts. Average text size is 44.1 characters.

We show two examples of actual extractions in Figures 5 and 6.

In the example shown in Figure 5, the prototype is able to recognize subtrees which have the same meaning despite having differently ordered dependencies e.g., *"Tantohsha-Ni Kaiyaku-Tetsudzuki-Wo Irai-Shita-Ga"* and *"Kaiyaku-No Tetsudzuki-Wo Tantohsha-Ni Irai-Shite"*. Such subtrees are unified by reordering their sister nodes. The two subtrees here would be unified into *"Kaiyaku-No Tetsudzuki-Wo Tantohsha-Ni Irai-Shita-Tokoro*".

In the extraction example shown in the Figure 6, the prototype is able to recognize subtrees which have the same meaning despite having different direction dependencies, e.g., *"Kanji-No Warui Taioh"* and *"Taioh-Ga Kanji-Ga Warui"*. Such subtrees are unified by root expansion. The two subtrees here would be unified into *"Taioh-Ga Kanji-Ga Warui"*.

We evaluated (1) the efficiency of extraction and (2) the efficiency of sentence construction, and the evaluation itself was based on a total of 900 characteristic subtrees extracted from Corpus 1 and 386 characteristic subtrees extracted from Corpus 2 that the prototype identified as having two or more nodes.

5.1 The Efficiency of Characteristic Subtree Extraction Based on the Use of Structural Transformations of Dependency Trees

Here, our relevant measures are the "precise extraction ratio," which represents the accuracy of subtree unification, and the "extraction coverage ratio," which represents the degree to which texts that should have yielded extraction results actually did yield them.

If we consider (1) *C* to represent those texts from

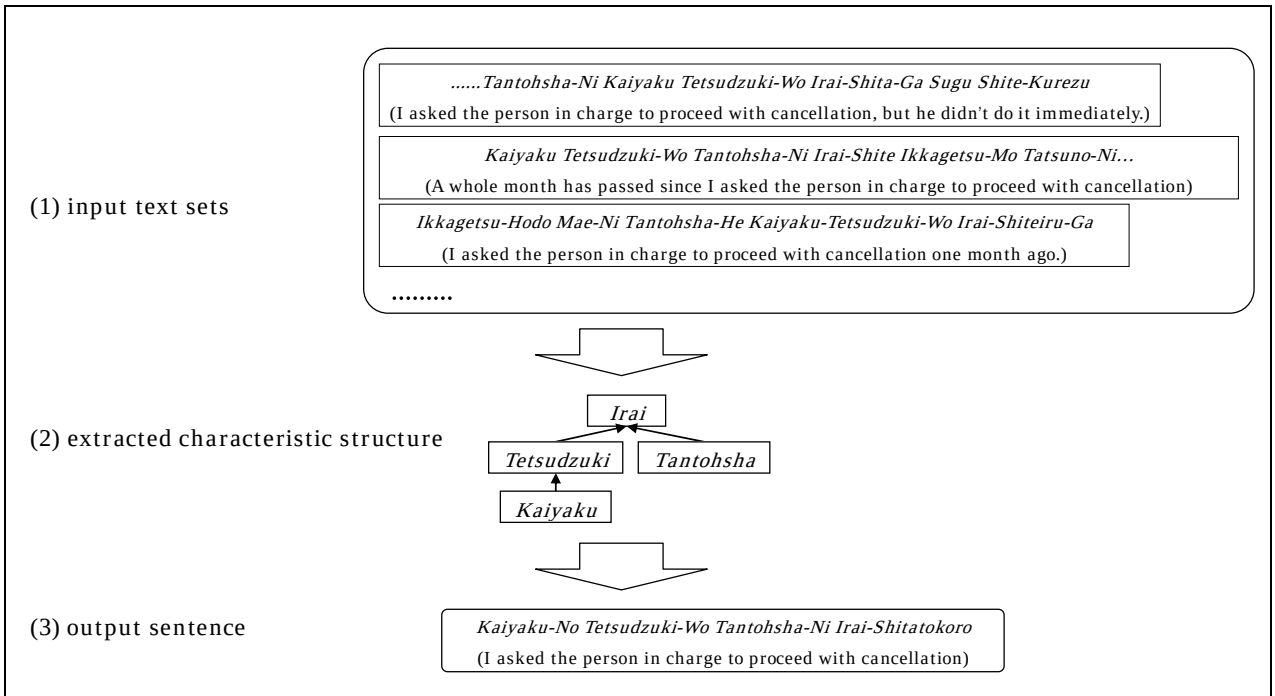


Figure 5: Extraction example 1.

which prototype unification and extraction resulted in characteristic subtrees having two or more nodes, (2) *B* to represent the texts from which human judgement would result in the identification of the same subtrees, and (3) *A* to represent the overlap of *B* and *C*, then the "precise extraction ratio" is *A*/*C*, and the "extraction coverage ratio" is *A*/*B*. Figure 7 shows the relationships among *A*, *B*, and *C*.

In order to identify the texts represented by the above *B*, we first conducted a search of all texts on the basis of sentences constructed by the prototype. The results of this search were then searched manually to determine which texts actually contained phrases expressing the same meanings as those expressed by the multi-node characteristic subtrees extracted by the prototype.

Table 1 lists both the precise extraction ratio and the extraction coverage ratio for each of the two corpuses.

corpus	precise extraction ratio	extraction coverage ratio
1	91.4%	82.5%
2	90.5%	84.4%

Table 1: Precise extraction ratio and extraction coverage ratio

As may be seen in the table, more than 90% of the texts identified by the prototype as containing

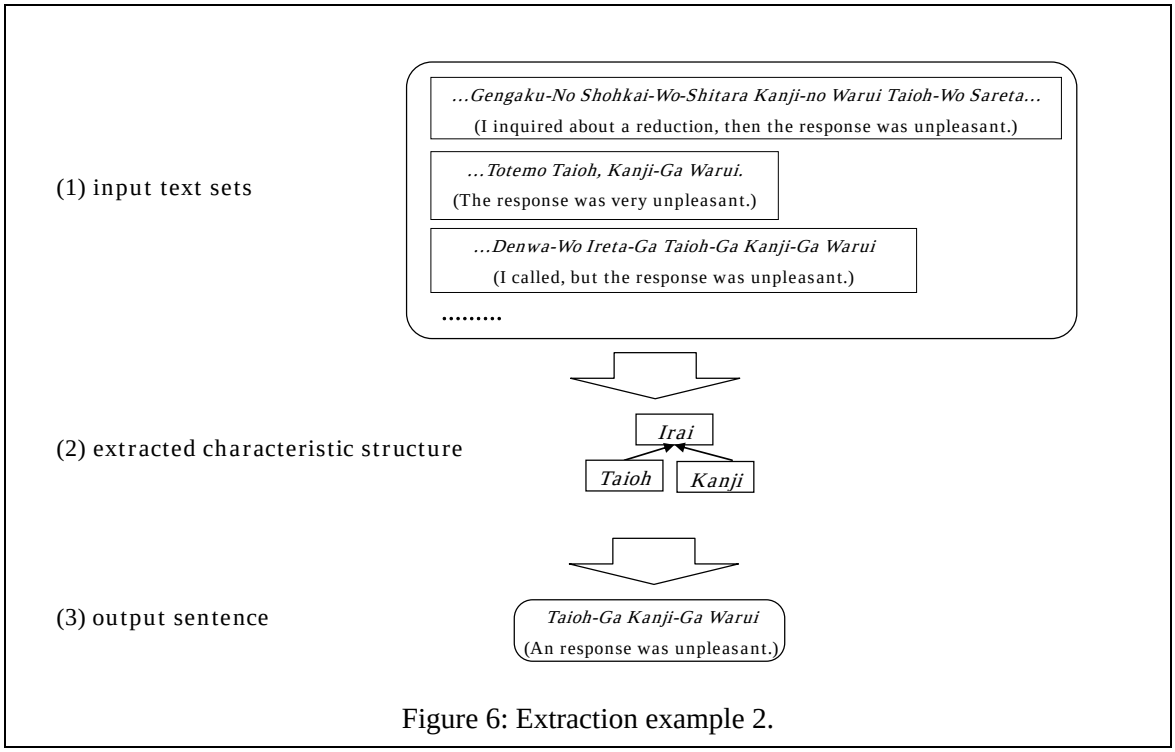
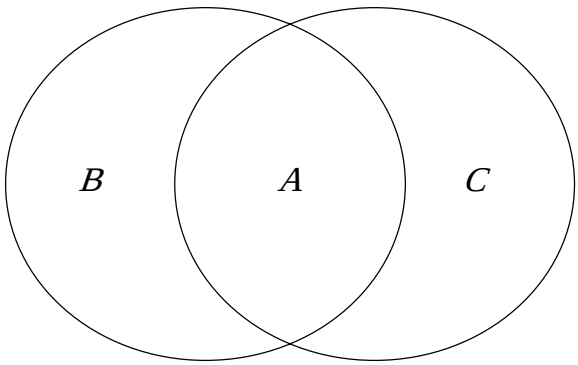


Figure 6: Extraction example 2.



A: Overlap of *B* and *C*.
B: The texts from which human judgement would result in the identification of the same subtrees.
C: The texts from which prototype unification and extraction resulted in characteristic subtrees having two or more nodes.

Figure 7: *A*, *B*, *C* in Section 5.1.

multi-node characteristic subtrees were also identified by human judgement as containing them. Further, more than 80% of those human-identified texts were also correctly identified by the prototype. This indicates a degree of accuracy and coverage sufficient for practical use. In addition, the reordering and the root expansion achieved a high level of reliability.

In texts which actually contained the meaning of a characteristic subtree but from which that characteristic subtree was not extracted, we found that most of those texts contained expressions which our structural transformation was unable to handle (For example, *"Pasuwahdo-Ga Wakarimasen"* (I don't know my password.) and *"Pasuwahdo-Wo Henkoh-shite Wakaranaku-Natte-Shimaimashita"* (I don't remember what I changed my password to.) were not identified as indicating the same basic problem).

5.2 The Efficiency of Sentence Construction from Characteristic Subtrees

In order to evaluate the efficiency of sentence construction from characteristic subtrees, we used an "extraction result comprehensibility ratio," which represents the ratio of the number of sentences whose meanings are comprehensible to the total number of sentences constructed from multi-node characteristic subtrees.

Further, because not all comprehensible sentences contain enough information to be meaningful for text mining purposes, we also employed a "meaningful extraction result ratio," which is the ratio of the number of sentences containing enough information to be meaningful to the total number of sentences constructed from multi-node subtrees.

Table 2 shows the above two ratios for each of the two corpuses.

corpus	extraction result comprehensibility ratio	meaningful extraction ratio
1	97.9%	87.8%
2	95.3%	83.9%

Table 2: Extraction result comprehensible ratio and meaningful extraction ratio

As may be seen in the table, more than 95% of the constructed sentences were comprehensible, and, on average, only about 10% of the comprehensible sentences failed to contain enough information to be meaningful for mining. These results also indicate the potential for practical applications.

Here, we should also note that most sentences judged to be incomprehensible had been constructed from wrongly parsed dependency trees.

With respect to comprehensible sentences which did not contain enough meaningful information, we were able to identify the following two significant tendencies:

Partial deletion of compound nouns: The partial deletion of one or more elements of a compound noun (e.g., the reduction of "*WEB Mail-Nite Mail-Wo Okuroh-To-Suru-To"* (When I tried to send an e-mail message by WEB mail, …) to *"Mail-Nite Mail-Wo Okuroh-To-Suru-To"* (When I tried to send an e-mail message by e-mail, …)) would be sufficient to render a sentence non-meaningful for the purposes of text mining.

Deletion of crucial element preceding declinable word: In some cases the deletion of an element preceding a declinable word (in Japanese, verbs and adjectival forms) can be enough to render a sentence non-meaningful for the purposes of text mining (e.g., the reduction of "*Dansa-Ga Ari Kiken-Da*" (There is danger due to a level-difference.) to "*Ari Kiken-Da*" (There is danger due to.)).

6 Conclusion and Future Work

In this paper, we have described a tree mining method for the efficient handling of differently ordered trees having the same meaning and for the outputting of easily understandable extraction results. With this method, different ordered trees which have the same meaning are transformed into a single unified ordered tree, and highly understandable results are produced by means of a newly developed technique for the construction of Japanese-language sentences from extracted characteristic subtrees. The application of a system prototype to a business report corpus and a contact-center corpus has demonstrated a precise extraction ratio of greater than 90% and an extraction coverage ratio of greater than 80%. This demonstrates the high reliability of both the reordering and the root expansion, and it indicates the potential for practical applications. Further, about 95% of the extracted sentences were comprehensible and about 85% contained enough information to be meaningful for the purposes of text mining.

In order to increase the extraction coverage ratio, we intend to introduce a structural replacement method capable of handling differences in tree ordering which cannot be handled by structural transformation. In order to reduce the amount of non-meaningful output, we intend to improve our method of handling both compound nouns and elements which precede declinable words and are crucial to meaningfulness.

References

T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto and S. Arikawa. 2002. Efficient substructure discovery from large semistructured data. *Proc. SDM'02*, pages 158–174.

T. Kudo and Y. Matsumoto. 2004. A Boosting Algorithm for Classification of Semi-Structured Text. *Proc. of EMNLP,* pages 301-308.

S. Morinaga, H. Arimura, T. Ikeda, Y. Sakao and S. Akamine, 2005, Key semantics extraction by dependency tree mining. *Proc. of KDDI2005 (to appear)*.