Japanese Word Reordering Integrated with Dependency Parsing

Kazushi Yoshida[1,a)]**Tomohiro Ohno**[2,b)] **Yoshihide Kato**[3,c)] **Shigeki Matsubara**[1,d)]
[1]Graduate School of Information Science, Nagoya University, Japan
[2]Information Technology Center, Nagoya University, Japan
[3]Information & Communications, Nagoya University, Japan
[a)]yoshida@db.ss.is.nagoya-u.ac.jp [b)]ohno@nagoya-u.jp
[c)]yoshihide@icts.nagoya-u.ac.jp [d)]matubara@nagoya-u.jp

Abstract

Although Japanese has relatively free word order, Japanese word order is not completely arbitrary and has some sort of preference. Since such preference is incompletely understood, even native Japanese writers often write Japanese sentences which are grammatically well-formed but not easy to read. This paper proposes a method for reordering words in a Japanese sentence so that the sentence becomes more readable. Our method can identify more suitable word order than conventional word reordering methods by concurrently performing dependency parsing and word reordering instead of sequentially performing the two processing steps. As the result of an experiment on word reordering using newspaper articles, we confirmed the effectiveness of our method.

1 Introduction

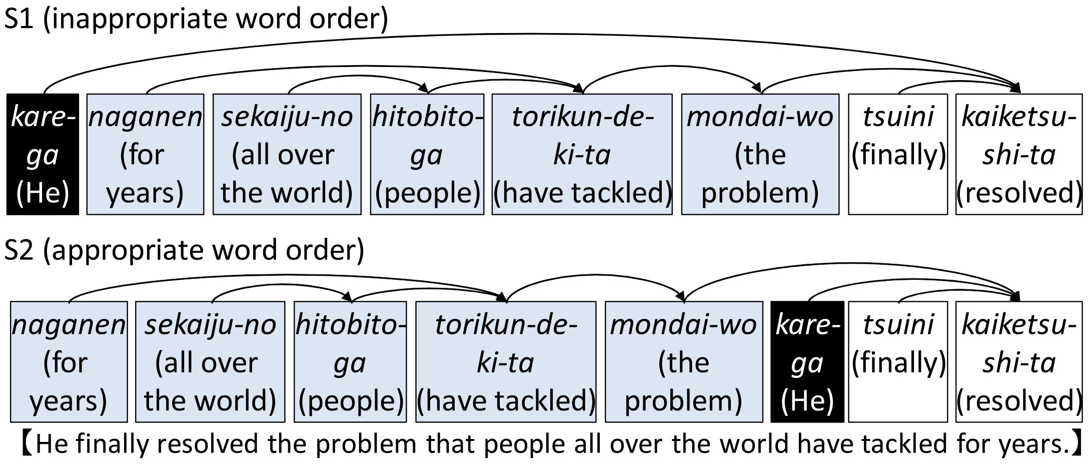
Japanese has relatively free word order, and thus Japanese sentences which make sense can be written without having a strong awareness of word order. However, Japanese word order is not completely arbitrary and has some sort of preference. Since such preference is incompletely understood, even native Japanese writers often write Japanese sentences which are grammatically well-formed but not easy to read. The word reordering of such sentences enables the readability to be improved.

There have been proposed some methods for reordering words in a Japanese sentence so that the sentence becomes easier to read (Uchimoto et al., 2000; Yokobayashi et al., 2004). In addition, there exist a lot of researches for estimating appropriate word order in various languages (Filippova and Strube, 2007; Harbusch et al., 2006; Kruijff et al., 2001; Ringger et al., 2004; Shaw and Hatzivassiloglou, 1999). Although most of these previous researches used syntactic information, the sentences they used there were what had been previously parsed. It is a problem that word reordering suffers the influence of parsing errors. Furthermore, as the related works, there are various researches on word reordering for improving the performance of statistical machine translation (Goto et al., 2012; Elming, 2008; Ge, 2010; Christoph and Hermann, 2003; Nizar, 2007). These researches consider information as to both a source language and a target language to handle word order differences between them. Therefore, their problem setting is different from that for improving the readability of a single language.

This paper proposes a method for reordering words in a Japanese sentence so that the sentence becomes easier to read for revision support. Our proposed method concurrently performs dependency parsing and word reordering for an input sentence of which the dependency structure is still unknown. Our method can identify more suitable word order than conventional word reordering methods because it can concurrently consider the preference of both word order and dependency. An experiment using newspaper articles showed the effectiveness of our method.

2 Word Order and Dependency in Japanese Sentences

There have been a lot of researches on Japanese word order in linguistics (for example, Nihongo Kijutsu Bunpo Kenkyukai, 2009; Saeki, 1998), which have marshalled fundamental contributing factors which



S1 (inappropriate word order)
kare-ga (He)
naganen (for years)
sekaiju-no (all over the world)
hitobito-ga (people)
torikun-de-ki-ta (have tackled)
mondai-wo (the problem)
tsuini (finally)
kaiketsu-shi-ta (resolved)
S2 (appropriate word order)
naganen (for years)
sekaiju-no (all over the world)
hitobito-ga (people)
torikun-de-ki-ta (have tackled)
mondai-wo (the problem)
kare-ga (He)
tsuini (finally)
kaiketsu-shi-ta (resolved)
【He finally resolved the problem that people all over the world have tackled for years.】

Figure 1: Example of inappropriate/appropriate word order

decide the appropriate word order in detail. In a Japanese sentence, a predicate of the main clause is fundamentally placed in last position, and thus, case elements, adverbial elements, or subordinate clauses are located before it. In addition, case elements are basically placed in the order of a nominative, a dative and an accusative. However, the basic order of case elements is often changed by being influenced from grammatical and discourse factors. For example, it is pointed out that a long case element has strong preference to be located at the beginning of a sentence even if the element is not nominative, as shown in Figure 1.

In Figure 1, a box and an arrow express a *bunsetsu*[1] and a dependency relation respectively. Both the sentences S1 and S2 have the same meaning which is translated as "He finally resolved the problem that people all over the world have tackled for years" in English. The difference between S1 and S2 is just in their word orders in Japanese.

The word order of S1 is more difficult to read than that of S2 because the distance between the bunsetsu "*kare-ga* (He)" and its modified bunsetsu "*kaiketsu-shi-ta* (resolved)" is large and thus the loads on working memory become large. This example suggests that if the dependency structure of S1 is identified, that information is useful to reorder the word order of S1 to that of S2 so that it becomes easier to read. In fact, most of the conventional word reordering methods have reordered words using the previously parsed dependency structure. However, the word order of S1 is thought to be more difficult to parse than that of S2 because dependency parsers are usually trained on syntactically annotated corpora in which sentences have the appropriate word order such as that in S2. This is why it is highly possible that dependency parsing can achieve a higher accuracy by changing the word order of S1 to that of S2 in advance.

The above observations indicate that word reordering and dependency parsing depend on each other. Therefore, we consider it is more desirable to concurrently perform the two processings than to sequentially perform them.

3 Word Reordering Method

In our method, a sentence, on which morphological analysis and bunsetsu segmentation have been performed, is considered as the input[2]. We assume that the input sentence might have unsuitable word order,

[1]*Bunsetsu* is a linguistic unit in Japanese that roughly corresponds to a basic phrase in English. A bunsetsu consists of one independent word and zero or more ancillary words. A dependency relation in Japanese is a modification relation in which a modifier bunsetsu depends on a modified bunsetsu. That is, the modifier bunsetsu and the modified bunsetsu work as modifier and modifyee, respectively.

[2]In order to focus attention on the comparison between our method and the conventional method, we assumed the input on which the lower layer processings than dependency parsing have been performed. Even if morphological analysis and bunsetsu segmentation are automatically performed on input sentences which have unsuitable word order, we can expect the accuracies

which is not easy to read but grammatically well-formed. Our method identifies the suitable word order which is easy to read by concurrently performing dependency parsing.

The simultaneous performing of dependency parsing and word reordering is realized by searching for the maximum-likelihood pattern of word order and dependency structure for the input sentence. Note our method reorders bunsetsus in a sentence without paraphrasing and does not reorder morphemes within a bunsetsu.

3.1 Probabilistic Model for Word Reordering

When a sequence of bunsetsus in an input sentence $B = b_1 \cdots b_n$ is provided, our method identifies the structure S which maximizes $P(S|B)$. The structure S is defined as a tuple $S = \langle O, D \rangle$ where $O = \{o_{1,2}, o_{1,3}, \cdots, o_{1,n}, \cdots, o_{i,j}, \cdots, o_{n-2,n-1}, o_{n-2,n}, o_{n-1,n}\}$ is the word order pattern after reordering and $D = \{d_1, \cdots, d_{n-1}\}$ is dependency structure. Here, $o_{i,j}$ $(1 \leq i < j \leq n)$ expresses the order between b_i and b_j after reordering. $o_{i,j}$ is 1 if b_i is located before b_j, and is 0 otherwise. In addition, d_i expresses the dependency relation whose modifier bunsetsu is b_i.

$P(S|B)$ for a $S = \langle O, D \rangle$ is calculated as follows.

$$\begin{aligned} P(S|B) &= P(O,D|B) \\ &= \sqrt{P(O|B) \times P(D|O,B) \times P(D|B) \times P(O|D,B)} \qquad (1) \end{aligned}$$

Formula (1) is obtained for the product of the following two formulas. According to the probability theory, the calculated result of Formula (1) is equal to those of Formulas (2) and (3). However, in practice, since each factor in the formulas is estimated based on the corpus used for training, the calculated results of these formulas are different from each other. We use Formula (1) to estimate $P(S|B)$ by using both values of $P(D|O,B)$ and $P(O|D,B)$. In fact, we pre-experimentally confirmed that the calculated result of Formula (1) was better than those of the others.

$$P(O,D|B) = P(O|B) \times P(D|O,B) \qquad (2)$$

$$P(O,D|B) = P(D|B) \times P(O|D,B) \qquad (3)$$

Assuming that order $o_{i,j}$ between two bunsetsus is independent of that between other two bunsetsus and that each dependency relation d_i is independent of the others, each factor in Formula (1) can be approximated as follows:

$$P(O|B) \cong \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} P(o_{i,j}|B) \qquad (4)$$

$$P(D|O,B) \cong \prod_{i=1}^{n-1} P(d_i|O,B) \qquad (5)$$

$$P(D|B) \cong \prod_{i=1}^{n-1} P(d_i|B) \qquad (6)$$

$$P(O|D,B) \cong \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} P(o_{i,j}|D,B) \qquad (7)$$

where $P(o_{i,j}|B)$ is the probability that the order between b_i and b_j is $o_{i,j}$ when B is provided, $P(d_i|O,B)$ is the probability that the dependency relation whose modifier bunsetsu is b_i is d_i when the sentence generated by reordering B according to O is provided, $P(d_i|B)$ is the probability that the dependency relation whose modifier bunsetsu is b_i is d_i when B is provided, and $P(o_{i,j}|D,B)$ is the probability that the order between b_i and b_j is $o_{i,j}$ when B where the dependency relation is D is provided. These probabilities are estimated by the maximum entropy method.

remain comparatively high. This is because their processings use mainly local information.

To estimate $P(d_i|O,B)$, we used the features used in Uchimoto et al. (1999) except when eliminating features about Japanese commas (called *toten*, which is a kind of punctuation) and quotation marks. To estimate $P(d_i|B)$, we used the features which can be obtained without information about the order of input bunsetsus among the features used in estimating $P(d_i|O,B)$. To estimate $P(o_{i,j}|D,B)$, if b_i and b_j modifies the same bunsetsu, we used the features used in Uchimoto et al. (2000), except when eliminating features about parallel relations and semantic features. Otherwise, we used the features left after eliminating features about modified bunsetsus from those used in the above-mentioned case. To estimate $P(o_{i,j}|B)$, we used the features which can be obtained without dependency information among the features used to estimate $P(O_{i,j}|D,B)$.

3.2 Search Algorithm

Since there are a huge number of the structures $S = \langle O, D \rangle$ which are theoretically possible for an input sentence B, an efficient algorithm is desired. However, since O and D are dependent on each other, it is difficult to find the optimal structure efficiently. In our research, we extend CYK algorithm used in conventional dependency parsing to efficiently find the suboptimal $S = \langle O, D \rangle$ which maximizes $P(S|B)$ efficiently.

Our research assumes that an input sentence, which is grammatically well-formed, is reordered without changing the meaning so that the sentence becomes much easier to read. From this assumption, we can use following conditions for efficient search:

1. The dependency structure of an input sentence should satisfy the following Japanese syntactic constraints under the input word order:
   - No dependency is directed from right to left.
   - Dependencies don't cross each other.
   - Each bunsetsu, except the last one, depends on only one bunsetsu.
2. Even after the words are reordered, the dependency structure should satisfy the above-mentioned Japanese syntactic constraints under the changed word order.
3. The dependency structures of a sentence before and after reordering should be identical.

Using the condition 1 and the condition 3, we can narrow down the search space of D to dependency structures that satisfy Japanese syntactic constraints under the input word order. Furthermore, the search space of O can be narrowed down to the word order patterns derived from the above narrowed dependency structures based on the conditions 2 and 3. That is, after dependency structures possible for an input sentence are narrowed down, we just have to find the word order patterns after reordering so that each of the dependency structures is maintained and satisfies the Japanese syntactic constraints even under the changed word order.

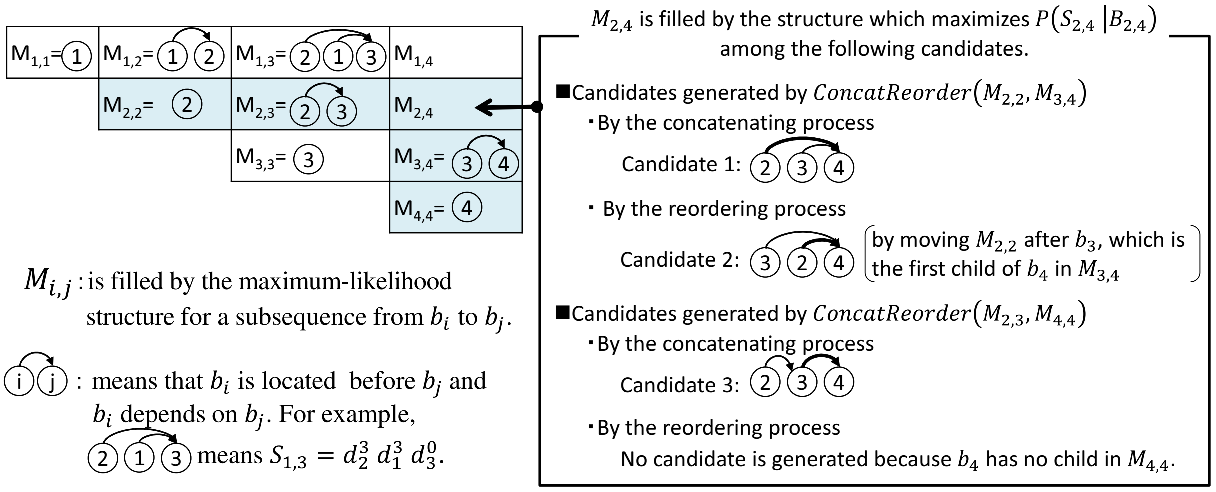
On the other hand, it is well known that CYK algorithm can efficiently find the optimal dependency structure which satisfies Japanese syntactic constraints. Therefore, in our research, we have extended the CYK algorithm for the conventional dependency parsing so that it can find the suboptimal D and O from among the dependency structures and word order patterns which satisfy the conditions 1, 2 and 3.

3.2.1 Word Reordering Algorithm

Algorithm 1 shows our word reordering algorithm. In our algorithm, the $n \times n$ triangular matrix $M_{i,j}(1 \leq i \leq j \leq n)$ such as the left-side figure in Figure 2 is prepared for an input sentence consisting of n numbers of bunsetsus. $M_{i,j}$, the element of the triangular matrix M in the i-th row and j-th column, is filled by $argmax_{S_{i,j}} P(S_{i,j}|B_{i,j})$, which is the maximum-likelihood structure for an input subsequence $B_{i,j} = b_i \cdots b_j$. In this section, for convenience of explanation, we represent $S_{i,j}$ as a sequence of dependency relations $d_x(i \leq x \leq j)$. For example, $S_{i,j} = d_i d_{i+1} \cdots d_j^0$ means that the first bunsetsu is b_i, the second is b_{i+1}, $\cdots$, the last is b_j, and the dependency structure is $\{d_i, d_{i+1}, \cdots, d_{j-1}\}$. Here, if we need to clearly specify the modified bunsetsu, we represent the dependency relation that bunsetsu

Algorithm 1 word reordering algorithm

```
1: input $B_{1,n} = b_1 \cdots b_n$ // input sentence
2: set $M_{i,j}$ $(1 \le i \le j \le n)$ // triangular matrix
3: set $C_{i,j}$ $(1 \le i \le j \le n)$ // set of structure candidates
4: for $i = 1$ to $n$ do
5:   $M_{i,i} = d_i^0$
6: end for
7: for $d = 1$ to $n - 1$ do
8:   for $i = 1$ to $n - d$ do
9:     $j = i + d$
10:    for $k = i$ to $j - 1$ do
11:      $C_{i,j} = C_{i,j} \cup ConcatReorder(M_{i,k}, M_{k+1,j})$
12:    end for
13:    $M_{i,j} = argmax_{S_{i,j} \in C_{i,j}} P(S_{i,j}|B_{i,j})$
14:  end for
15: end for
16: return $M_{1,n}$
```



M1,1=1
M1,2=1 2
M1,3=2 1 3
M1,4
M2,2= 2
M2,3=2 3
M2,4
M3,3= 3
M3,4= 3 4
M4,4= 4
$M_{i,j}$: is filled by the maximum-likelihood structure for a subsequence from b_i to b_j.
i j : means that b_i is located before b_j and b_i depends on b_j. For example, 2 1 3 means $S_{1,3} = d_2^3\ d_1^3\ d_3^0$.
$M_{2,4}$ is filled by the structure which maximizes $P(S_{2,4}|B_{2,4})$ among the following candidates.
■Candidates generated by $ConcatReorder(M_{2,2}, M_{3,4})$
・By the concatenating process
Candidate 1: 2 3 4
・ By the reordering process
Candidate 2: 3 2 4
by moving $M_{2,2}$ after b_3, which is the first child of b_4 in $M_{3,4}$
■Candidates generated by $ConcatReorder(M_{2,3}, M_{4,4})$
・By the concatenating process
Candidate 3: 2 3 4
・By the reordering process
No candidate is generated because b_4 has no child in $M_{4,4}$.

Figure 2: Execution example of our search algorithm

b_x modifies b_y as d_x^y. In addition, d_j^0 means that the last bunsetsu of the subsequence don't modify any bunsetsu.

First, the statements of the lines 4 to 6 fill each of diagonal elements $M_{i,i}$ $(1 \le i \le n)$ with d_i^0. Next, the statements of the lines 7 to 15 fill $M_{i,j}$ in turn toward the upper right $M_{1,n}$ along the diagonal line, starting from the diagonal elements $M_{i,i}$. The maximum-likelihood structure which should fill an $M_{i,j}$ is found as follows:

The statements of the lines 10 to 12 repeat the process of generating candidates of the maximum-likelihood structure from $M_{i,k}$ and $M_{k+1,j}$ by the function $ConcatReorder$, and adding them to the set of structure candidates $C_{i,j}$. The function $ConcatReorder$ takes two arguments of $M_{i,k}$ and $M_{k+1,j}$ and returns the set of candidates of the maximum-likelihood structure which should fill $M_{i,j}$. The function $ConcatReorder$ is composed of two processes: concatenating process and reordering process. First, the concatenating process generates a candidate by simply concatenating $M_{i,k}$ and $M_{k+1,j}$ in turn about the word order and connecting $M_{i,k}$ and $M_{k+1,j}$ by the dependency relation between the last bunsetsus of them about the dependency structure, without changing the internal structure of each of them. For example, when $M_{i,k} = d_i d_{i+1} \cdots d_{k-1} d_k^0$ and $M_{k+1,j} = d_{k+1} d_{k+2} \cdots d_{j-1} d_j^0$ are given as the argument, the concatenating process generates "$d_i d_{i+1} \cdots d_{k-1} d_k^j d_{k+1} d_{k+2} \cdots d_{j-1} d_j^0$."

Second, the reordering process generates candidates by reordering words in the candidate generated by the concatenating process. The reordering is executed on the following conditions. The first condition is that the dependency structure is maintained and satisfies the Japanese syntactic constraints even under the changed word order. The second condition is that the order of any two words within each of $M_{i,k}$ and $M_{k+1,j}$ is maintained. Concretely, the first reordered candidate is generated by moving $M_{i,k}$ after the first (leftmost) child[3] of the last bunsetsu of $M_{k+1,j}$ among the children in $M_{k+1,j}$. Then, the second reordered candidate is generated by moving $M_{i,k}$ after the second child. The reordering process is continued until the last reordered candidate is generated by moving $M_{i,k}$ after the last child. That is, the number of candidates generated by the reordering process is equal to the number of children of the last bunsetsu in $M_{k+1,j}$. For example, when $M_{i,k} = d_i d_{i+1} \cdots d_{k-1} d_k^0$ and $M_{k+1,j} = d_{k+1}^j d_{k+2}^j \cdots d_{j-1}^j d_j^0$, which means all bunsetsus except the last one depend on the last one, are given, the reordering process generates the following $j-k-1$ candidates: "$d_{k+1}^j d_i d_{i+1} \cdots d_{k-1} d_k^j d_{k+2}^j d_{k+3}^j \cdots d_{j-1}^j d_j^0$," "$d_{k+1}^j d_{k+2}^j d_i d_{i+1} \cdots d_{k-1} d_k^j d_{k+3}^j d_{k+4}^j \cdots d_{j-1}^j d_j^0$," ..., and "$d_{k+1}^j d_{k+2}^j \cdots d_{j-1}^j d_i d_{i+1} \cdots d_{k-1} d_k^j d_j^0$." Therefore, in this case, the function $ConcatReorder$ finally returns the set of candidates of which size is $j-k$, which includes the candidates generated by the reordering process and a candidate generated by the concatenating process. Next, in the line 13, our algorithm fills in $argmax_{S_{i,j} \in C_{i,j}} P(S_{i,j}|B_{i,j})$ which is the maximum-likelihood structure for a subsequence $B_{i,j}$ on $M_{i,j}$.

Finally, our algorithm outputs $M_{1,n}$ as the maximum-likelihood structure of word order and dependency structure for the input sentence.

Note that if the function *ConcatReorder* is changed to the function *Concat* in the line 11, our algorithm becomes the same as CYK algorithm used in the conventional dependency parsing. The function $Concat$ takes two arguments of $M_{i,k}$ and $M_{k+1,j}$ and generates a candidate of the maximum-likelihood structure which should fill $M_{i,j}$ by the same way as the concatenating process in the function $ConcatReorder$. Then, the function $Concat$ returns the set which has the generated candidate as a element, of which size is 1.

3.2.2 Execution Example of Word Reordering Algorithm

Figure 2 represents an example of execution of our word reordering algorithm in $n = 4$. The left side of Figure 2 represents the triangle diagram which has 4×4 dimensions. The elements of the triangle diagram $M_{1,1}, M_{2,2}, M_{3,3}, M_{4,4}, M_{1,2}, M_{2,3}, M_{3,4}$, and $M_{1,3}$ have already been filled in turn, and $M_{2,4}$ is being filled. The right side of Figure 2 shows the process of calculating the maximum-likelihood structure which should fill $M_{2,4}$. First, in the loop from the line 10 to the line 12 in Algorithm 1, two structure candidates are generated by $ConcatReorder(M_{2,2}, M_{3,4})$. The candidate 1 is generated by the concatenating process, that is, by simply concatenating $M_{2,2}$ and $M_{3,4}$ and connecting the last bunsetsu of $M_{2,2}$ and that of $M_{3,4}$. The candidate 2 is generated by the reordering process, that is, by moving $M_{2,2}$ after b_3, which is the first child of b_4 in $M_{3,4}$. Second, the candidate 3 is generated by the concatenating process in $ConcatReorder(M_{2,3}, M_{4,4})$. On the other hand, the reordering process in $ConcatReorder(M_{2,3}, M_{4,4})$ generates no candidates because b_4 has no child in $M_{4,4}$. Among the three structures generated in the above way, the structure which maximizes $P(S_{2,4}|B) = P(O_{2,4}, D_{2,4}|B_{2,4})$ fills $M_{2,4}$.

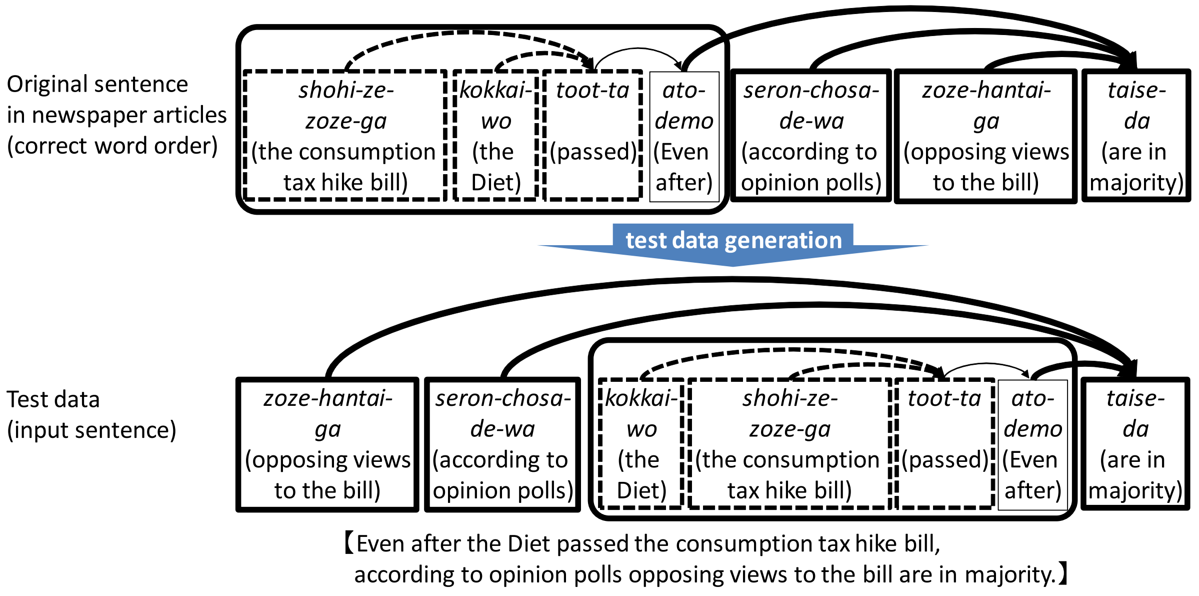
4 Experiment

To evaluate the effectiveness of our method, we conducted an experiment on word reordering by using Japanese newspaper articles.

4.1 Outline of Experiment

In the experiment, as the test data, we used sentences generated by only changing the word order of newspaper article sentences in Kyoto Text Corpus (Kurohashi and Nagao, 1998), maintaining the dependency structure. That is, we artificially generated sentences which made sense but were not easy to read,

[3]When b_i depends on b_j, we call b_i as a child of b_j. Furthermore, if b_j has more than or equal to one child, the children are numbered from left to right based on their positions.



Original sentence in newspaper articles (correct word order)
shohi-ze-zoze-ga (the consumption tax hike bill)
kokkai-wo (the Diet)
toot-ta (passed)
ato-demo (Even after)
seron-chosa-de-wa (according to opinion polls)
zoze-hantai-ga (opposing views to the bill)
taise-da (are in majority)
test data generation
Test data (input sentence)
zoze-hantai-ga (opposing views to the bill)
seron-chosa-de-wa (according to opinion polls)
kokkai-wo (the Diet)
shohi-ze-zoze-ga (the consumption tax hike bill)
toot-ta (passed)
ato-demo (Even after)
taise-da (are in majority)
【Even after the Diet passed the consumption tax hike bill, according to opinion polls opposing views to the bill are in majority.】

Figure 3: Example of test data generation

in order to focus solely on problems caused by unsuitable word order. Figure 3 shows an example of the test data generation. The generation procedure is as follows:

1. Find a bunsetsu modified by multiple bunsetsus from the sentence end.
2. Change randomly the order of the sub-trees which modify such bunsetsu.
3. Iterate 1 and 2 until reaching the beginning of the sentence.

In Figure 3, the bunsetsus "*taise-da* (are in the majority)" and "*toot-ta* (passed)" are found as bunsetsus modified by multiple bunsetsus. For example, when "*toot-ta* (passed)" is found, the order of "*shohi-ze-zoze-ga* (the consumption tax hike bill)" and "*kokkai-wo* (the Diet)" is randomly changed. In this experiment, all Japanese commas (*toten*) in a sentence, and sentences which have quotation marks were removed.

In this way, we artificially generated 865 sentences (7,620 bunsetsus) from newspaper articles of Jan. 9 in Kyoto Text Corpus and used them as the test data. As the training data, we used 7,976 sentences in 7 days' newspaper articles (Jan. 1, 3-8). Here, we used the maximum entropy method tool (Zhang, 2008) with the default options except "-i 1000."

In the evaluation of word reordering, we obtained the following two measurements, which are defined by Uchimoto et al. (2000):

- **complete agreement:** the percentage of the sentences in which all words' order completely agrees with that of the original sentence.
- **pair agreement:** the percentage of the pairs of bunsetsus whose word order agrees with that in the original sentence. (For example, in Figure 3, if the word order of the input sentence is not changed after reordering, the pair agreement is 52.4% ($= 11/{}_7C_2$) because the 11 pairs out of the ${}_7C_2$ pairs are the same as those in the original sentence.)

In the evaluation of dependency parsing, we obtained the **dependency accuracy** (the percentage of correctly analyzed dependencies out of all dependencies) and **sentence accuracy** (the percentage of the sentences in which all the dependencies are analyzed correctly), which are defined by Sekine et al. (2000).

For comparison, we established two baselines. Both of the baselines execute the dependency parsing primarily, and then, perform the word reordering by using the conventional word reordering method

Table 1: Experimental results (word reordering)

	pair agreement	complete agreement
our method	77.3% (30,190/38,838)	25.7% (222/865)
baseline 1	75.4% (29,279/38,838)*	23.8% (206/865)
baseline 2	74.8% (29,067/38,838)*	23.5% (203/865)
no reordering	61.5% (23,886/38,838)*	8.0% (69/865)*

Note that the agreements followed by * differ significantly from those of our method ($p < 0.05$).

Table 2: Experimental results (dependency parsing)

	dependency accuracy	sentence accuracy
our method	78.4% (5,293/6,755)	35.3% (305/865)
baseline 1	79.2% (5,350/6,755)	31.6% (273/865)*
baseline 2	81.2% (5,487/6,755)*	32.1% (278/865)*

Note that the accuracies followed by * differ significantly from those of our method ($p < 0.05$).

(Uchimoto et al., 1999). The difference between the two is the method of dependency parsing. The baselines 1 and 2 use the dependency parsing method proposed by Uchimoto et al. (2000) and the dependency parsing tool CaboCha (Kudo and Matsumoto, 2002), respectively. The features used for the word reordering in both the baselines are the same as those used to estimate $P(o_{i,j}|D,B)$ in our method. Additionally, the features used for the dependency parsing in the baseline 1 are the same as those used to estimate $P(d_i|O,B)$ in our method.

4.2 Experimental Results

Table 1 shows the experimental results on word reordering of our method and the baselines. Here, the last row shows the agreements measured by comparing the input word order with the correct word order. The agreements mean the values which can be achieved with no reordering[4]. The pair and complete agreements of our method were highest among all. The pair agreement of our method is significantly different from those of both the baselines ($p < 0.05$) although there is no significant difference between the complete agreements of them.

Next, Table 2 shows the experimental results on dependency parsing. The sentence accuracy of our method is significantly higher than those of both the baselines ($p < 0.05$). On the other hand, the dependency accuracy of our method is significantly lower than that of the baseline 2 although there is no significant difference between the dependency accuracies of our method and the baseline 1 ($p > 0.05$). Here, if the input sentences had the correct word order, the dependency accuracies of the baselines 1 and 2 were 86.4% (5,835/6,755) and 88.1% (5,950/6,755), respectively. We can see that the unsuitable word order caused a large decrease of the accuracies of the conventional dependency parsing methods. This is why the word order agreements of the baselines were decreased.

Figure 4 shows an example of sentences of which all bunsetsus were correctly reordered and the dependency structure was correctly parsed only by our method. We can see that our method can achieve the complicated word reordering. On the other hand, Figure 5 shows an example of sentences incorrectly reordered and parsed by our method. In this example, our method could not identify the correct modified bunsetsu and the appropriate position of the bunsetsu "*arikata-wo* (whole concept)." This is because the dependency probability between the bunsetsu "*arikata-wo* (whole concept)" and the bunsetsu "*fukume*

[4]Some input sentences were in complete agreement with the original ordering. There were some cases that the randomly reordered sentences accidentally have the same word order as the original ones. In addition, there were some sentences in which all bunsetsus except the last one depend on the next bunsetsu. The word order of such sentences is not changed by the test data generation procedure because the procedure is executed on condition of maintaining the dependency structure.

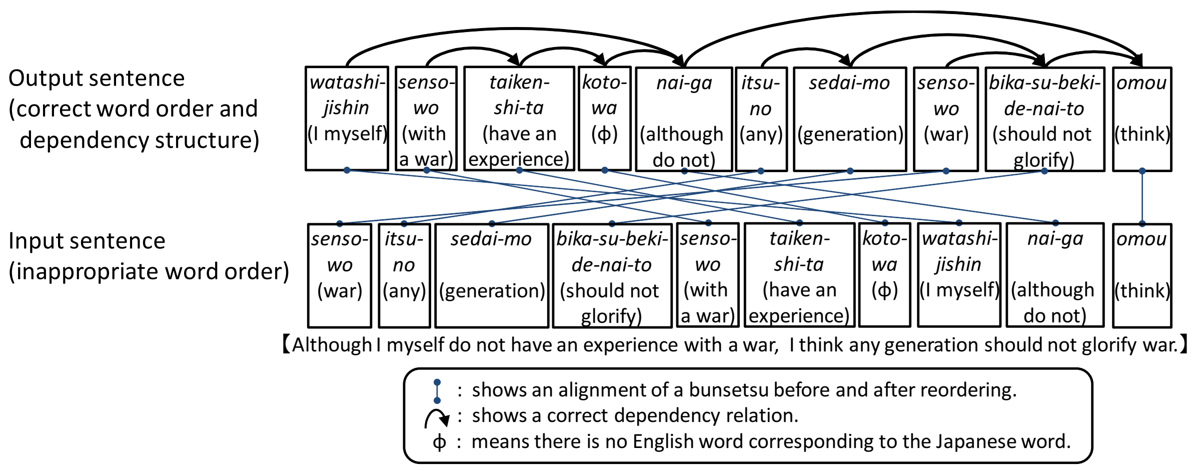

Output sentence (correct word order and dependency structure)
watashi-jishin (I myself)
senso-wo (with a war)
taiken-shi-ta (have an experience)
koto-wa (ϕ)
nai-ga (although do not)
itsu-no (any)
sedai-mo (generation)
senso-wo (war)
bika-su-beki-de-nai-to (should not glorify)
omou (think)
Input sentence (inappropriate word order)
senso-wo (war)
itsu-no (any)
sedai-mo (generation)
bika-su-beki-de-nai-to (should not glorify)
senso-wo (with a war)
taiken-shi-ta (have an experience)
koto-wa (ϕ)
watashi-jishin (I myself)
nai-ga (although do not)
omou (think)
【Although I myself do not have an experience with a war, I think any generation should not glorify war.】
: shows an alignment of a bunsetsu before and after reordering.
: shows a correct dependency relation.
ϕ : means there is no English word corresponding to the Japanese word.

Figure 4: Example of sentences correctly reordered and parsed by our method

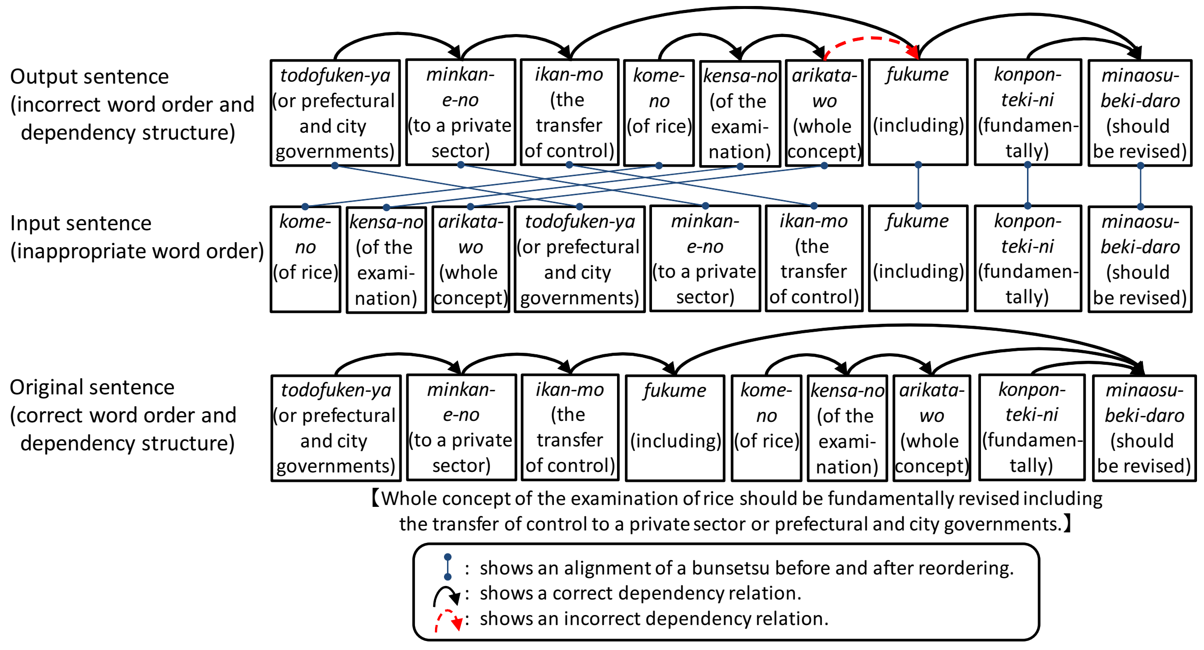

Output sentence (incorrect word order and dependency structure)
todofuken-ya (or prefectural and city governments)
minkan-e-no (to a private sector)
ikan-mo (the transfer of control)
kome-no (of rice)
kensa-no (of the exami-nation)
arikata-wo (whole concept)
fukume (including)
konpon-teki-ni (fundamen-tally)
minaosu-beki-daro (should be revised)
Input sentence (inappropriate word order)
kome-no (of rice)
kensa-no (of the exami-nation)
arikata-wo (whole concept)
todofuken-ya (or prefectural and city governments)
minkan-e-no (to a private sector)
ikan-mo (the transfer of control)
fukume (including)
konpon-teki-ni (fundamen-tally)
minaosu-beki-daro (should be revised)
Original sentence (correct word order and dependency structure)
todofuken-ya (or prefectural and city governments)
minkan-e-no (to a private sector)
ikan-mo (the transfer of control)
fukume (including)
kome-no (of rice)
kensa-no (of the exami-nation)
arikata-wo (whole concept)
konpon-teki-ni (fundamen-tally)
minaosu-beki-daro (should be revised)
【Whole concept of the examination of rice should be fundamentally revised including the transfer of control to a private sector or prefectural and city governments.】
: shows an alignment of a bunsetsu before and after reordering.
: shows a correct dependency relation.
: shows an incorrect dependency relation.

Figure 5: Example of sentences incorrectly reordered and parsed by our method

(including)" is higher than the one between the bunsetsu "*arikata-wo* (whole concept)" and the bunsetsu "*minaosu-beki-daro* (should be revised)", and the probability that the bunsetsu "*arikata-wo* (whole concept)" is located at the left side of "*fukume* (including)" is higher than that of the right side. Since the word order of the output sentence has a strong probability of causing a wrong interpretation like "The transfer of control to a private sector or prefectural and city governments should be fundamentally revised including whole concept of the examination of rice.", this reordering has a harmful influence on the comprehension. We need to study techniques for avoiding the word order which causes the change of meanings in an input sentence.

From the above, we confirmed the effectiveness of our method on word reordering and dependency parsing of a sentence of which the word order is not easy to read.

5 Conclusion

This paper proposed the method for reordering bunsetsus in a Japanese sentence. Our method can identify suitable word order by concurrently performing word reordering and dependency parsing. Based on the idea of limiting the search space using the Japanese syntactic constraints, we made the search algorithm by extending the CYK algorithm used in the conventional dependency parsing, and found the optimal structure efficiently. The result of the experiment using newspaper articles showed the effectiveness of our method.

In our future works, we would like to collect sentences written by Japanese subjects who do not have much writing skills, to conduct an experiment using those sentences. In addition, we would like to conduct a subjective evaluation to investigate whether the output sentences are indeed more readable than the input ones.

Acknowledgments

This research was partially supported by the Grant-in-Aid for Young Scientists (B) (No.25730134) and Challenging Exploratory Research (No.24650066) of JSPS.

References

Tillmann Christoph and Ney Hermann. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29(1):97–133.

Jakob Elming. 2008. Syntactic reordering integrated with phrase-based SMT. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING2008)*, pages 209–216.

Katja Filippova and Michael Strube. 2007. Generating constituent order in German clauses. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL2007)*, pages 320–327.

Niyu Ge. 2010. A direct syntax-driven reordering model for phrase-based machine translation. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT2010)*, pages 849–857.

Geert-Jan M. Kruijff, Ivana Kruijff-Korbayová, John Bateman, and Elke Teich. 2001. Linear order as higher-level decision: Information structure in strategic and tactical generation. In *Proceedings of the 8th European Workshop on Natural Language Generation (ENLG2001)*, pages 74–83.

Isao Goto, Masao Utiyama, and Eiichiro Sumita. 2012. Post-ordering by parsing for Japanese-English statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL2012)*, pages 311–316.

Karin Harbusch, Gerard Kempen, Camiel van Breugel, and Ulrich Koch. 2006. A generation-oriented workbench for performance grammar: Capturing linear order variability in German and Dutch. In *Proceedings of the 4th International Natural Language Generation Conference (INLG2006)*, pages 9–11.

Nihongo Kijutsu Bunpo Kenkyukai, editor. 2009. *Gendai nihongo bunpo 7 (Contemporary Japanese Grammar 7)*, pages 165–182. *Kuroshio Shuppan*. (In Japanese).

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of the 6th Conference on Computational Natural Language Learning (CoNLL2002)*, pages 63–69.

Sadao Kurohashi and Makoto Nagao. 1998. Building a Japanese parsed corpus while improving the parsing system. In *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC '98)*, pages 719–724.

Habash Nizar. 2007. Syntactic preprocessing for statistical machine translation. In *Proceedings of the 11th Machine Translation Summit (MT SUMMIT XI)*, pages 215–222.

Eric Ringger, Michael Gamon, Robert C. Moore, David Rojas, Martine Smets, and Simon Corston-Oliver. 2004. Linguistically informed statistical models of constituent structure for ordering in sentence realization. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING2004)*, pages 673–679.

Tetsuo Saeki. 1998. *Yosetsu nihonbun no gojun (Survey: Word Order in Japanese Sentences). Kuroshio Shuppan*. (In Japanese).

Satoshi Sekine, Kiyotaka Uchimoto, and Hitoshi Isahara. 2000. Backward beam search algorithm for dependency analysis of Japanese. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING2000)*, volume 2, pages 754–760.

James Shaw and Vasileios Hatzivassiloglou. 1999. Ordering among premodifiers. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL '99)*, pages 135–143.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 1999. Japanese dependency structure analysis based on maximum entropy models. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL '99)*, pages 196–203.

Kiyotaka Uchimoto, Masaki Murata, Qing Ma, Satoshi Sekine, and Hitoshi Isahara. 2000. Word order acquisition from corpora. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING2000)*, volume 2, pages 871–877.

Hiroshi Yokobayashi, Akira Suganuma, and Rin-ichiro Taniguchi. 2004. Generating candidates for rewriting based on an indicator of complex dependency and it's application to a writing tool. *Journal of Information Processing Society of Japan*, 45(5):1451–1459. (In Japanese).

Le Zhang. 2008. Maximum entropy modeling toolkit for Python and C++. `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`. [Online; accessed 1-March-2008].