Word Reordering and Comma Insertion Integrated with Shift-Reduce Dependency Parsing

Kota Miyachi[1], Tomohiro Ohno[2][a] and Shigeki Matsubara[3][b]

[1]*Graduate School of Informatics, Nagoya University, Nagoya, Japan*

[2]*Graduate School of Advanced Science and Technology, Tokyo Denki University, Tokyo, Japan*

[3]*Information & Communications, Nagoya University, Nagoya, Japan*

Keywords: Sentence Generation, Parsing, Paraphrasing, Readability, Punctuation Insertion.

Abstract: Japanese has widely recognized as relatively free word order language. However, since Japanese word order is not completely arbitrary and has some sort of preference, even native Japanese writers often write Japanese sentences that are grammatically well-formed but not easy to read. Furthermore, in Japanese sentences, a comma plays an important role in explicitly separating the constituents such as words and phrases. This paper proposes a method of word reordering and comma insertion for hard-to-read Japanese sentences so that they become easier to read, as basic technique. Our contribution is to show the feasibility of concurrently performing word reordering, comma insertion, and dependency parsing.

1 INTRODUCTION

Japanese has a relatively free word order structure. This characteristic makes the possibility of writing meaningful sentences without paying much attention to word order. In practice, however, the existence of word order preferences can lead to grammatically correct but hard-to-read sentences. Similarly, commas are relatively freely inserted into a Japanese sentence, but there are preferences, so it is necessary to place commas in the right places to make a sentence easy to read.

Reordering of words, which arranges a sentence into readable word order, has been studied as a basic technique for several applications such as writing assistance and sentence generation. Yoshida et al. (2014) proposed a word reordering method by concurrently executing it with dependency parsing. However, in this method, commas are not considered. On the other hand, Murata et al. (2010) developed a Japanese comma insertion method by analyzing features which capture the tendency of commas appearing in Japanese texts. Since this method uses dependency information, if the word order of an input sentence is hard-to-read, the accuracy of comma insertion would decrease. The reason is that an increase of errors of dependency parsing affect the accuracy.

In this paper, we propose a method of simultaneous execution of dependency parsing, word reordering and comma insertion for hard-to-read Japanese sentences. With their simultaneous execution, we can analyze the dependencies and comma positions of an input sentence considering not only the word order of the input sentence but also the sequence of words after reordering.

This paper is organized as follows: Section 2 explains aspects of word order, comma, and dependency in Japanese. In Section 3, we propose a method of word reordering and comma insertion integrated with shift-reduce dependency parsing, while Section 4 conducts an experiment to evaluate the proposed method quantitatively. Finally, Section 5 summarizes the paper.

2 WORD ORDER, COMMA, DEPENDENCY IN JAPANESE

The word order in a sentence relates to dependencies of the sentence. Uchimoto et al. (2000) have conducted word reordering using the syntactic information based on the premise that dependency parsing has been preliminarily performed, as well as (Belz and Kow, 2011; Filippova and Strube, 2007; Har-

[a] https://orcid.org/0000-0001-7015-7714
[b] https://orcid.org/0000-0003-0416-3635

Miyachi, K., Ohno, T. and Matsubara, S.
Word Reordering and Comma Insertion Integrated with Shift-Reduce Dependency Parsing.
DOI: 10.5220/0010335911441150
In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021) - Volume 2*, pages 1144-1150
ISBN: 978-989-758-484-8
Copyright © 2021 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

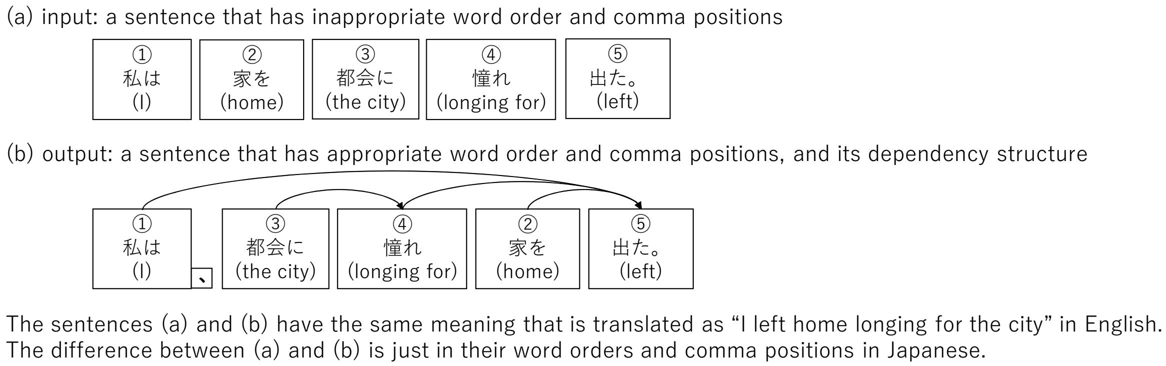

Figure 1: An example of the input and its ideal output in our method.

busch et al., 2006; Kruijff et al., 2001; Ringger et al., 2004; Shaw and Hatzivassiloglou, 1999; Yokobayashi et al., 2004). However, since dependency parsers are usually trained on syntactically annotated corpora in which sentences have the appropriate word order (e.g., newspaper articles sentences); the accuracy of dependency parsing is reduced when the input sentences are not easy to read because of their word orders. For the same reason, comma insertion is also reduced.

On the other hand, if the word reordering is conducted before the dependency parsing, the accuracy of the word order would be decrease because the dependency information cannot be used. In addition, since comma information is used in dependency parsing, the accuracy of the dependency parsing might be decreased if dependency parsing is performed before comma insertion.

As mentioned above, it can be concluded that dependency parsing, word reordering, and comma insertion are interdependent. Therefore, as an approach to the tasks of word reordering and comma insertion for hard-to-read sentences due to the word order, it seems suitable to perform them concurrently.

3 PROPOSED METHOD

In our method, we assume that the input sentence might have inappropriate word order and comma positions, which causes not easy to read it but grammatically well-formed (e.g., Figure 1(a)). Based on the assumption, our method simultaneously identifies the dependency structure, appropriate word order and comma positions in the sentence (e.g., Figure 1(b)).

As one of the strategies for executing these three processes simultaneously, we can think of extending (Yoshida et al., 2014), which achieved simultaneous execution of dependency parsing and word reordering, by concurrently performing also comma insertion. In other words, the strategy is to search for the most likely pattern among all possible patterns of dependency structure, word order, and comma positions for a whole sentence using dynamic programming. However, adopting such a strategy needs to consider all possible patterns, including comma positions, along with the dependency structure and word order. Due to this fact, the potential number of such patterns is too huge. Basides, it is not easy to search for maximum likelihood patterns efficiently because dynamic programming cannot be simply applied due to the non-independent nature of each process.

Our method realizes the simultaneous execution of dependency parsing, word reordering, and comma insertion by focusing on two local bunsetsus [1] in the input Japanese sentence, and determining the dependency, word order, and existence or non-existence of a comma between the two local bunsetsus.

3.1 Algorithm

In this study, we extend the Shift-Reduce algorithm for Japanese dependency parsing (Sassano, 2004) to realize the strategy of simultaneously determining the dependency, word order, and comma between two local bunsetsus in an input sentence.

The Shift-Reduce algorithm (Sassano, 2004) determines the dependency structure of a whole sentence by repeating the choice of two operators, Shift and Reduce, depending on whether the bunsetsu in the top of the stack depends on the bunsetsu in the front of the queue or not. In this study, we extended the algorithm by adding new operators for comma in-

[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

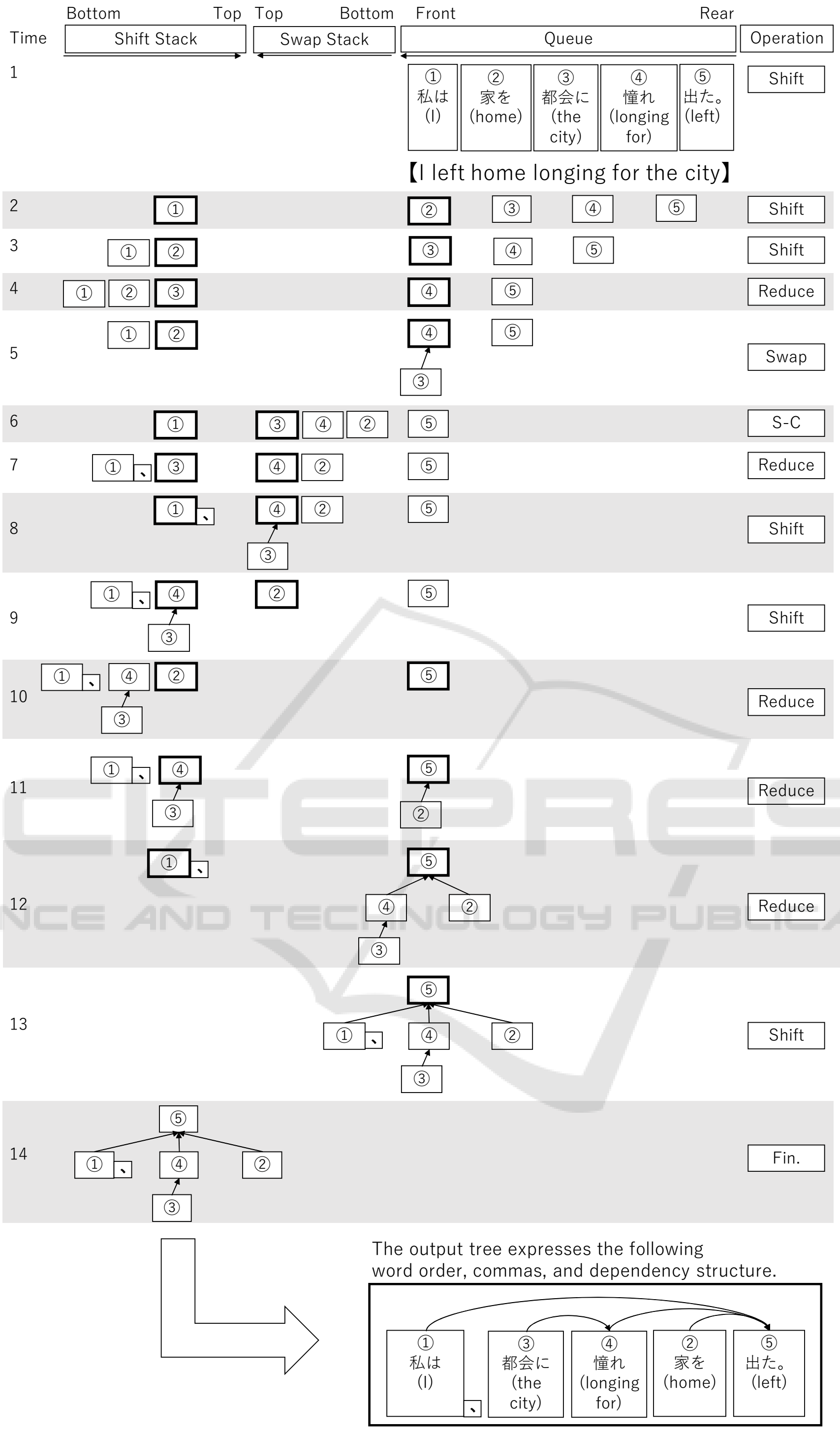

Figure 2: An example of simultaneous execution process of dependency parsing, word reordering, and comma insertion.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

sertion (Shift-Comma and Reduce-Comma) and word reordering (Swap) and a new stack for word reordering (swap stack), thus realized simultaneous execution of dependency parsing, word reordering, and comma insertion. In the following, the stack originally used in the Shift-Reduce algorithm (Sassano, 2004) is called simply "shift stack" to distinguish it from a new stack for word reordering (swap stack).

Figure 2 shows the simultaneous execution of dependency parsing, word reordering, and comma insertion in our method. Our method processes the bunsetsu sequence of an input sentence in order from the begining to as follows:

1. We store the bunsetsu sequence of an input sentence in the input order in the queue, and make both the shift and swap stacks empty.
2. It is repeated that one of five operators (Shift, Shift-Comma, Reduce, Reduce-Comma and Swap) is selected to manipulate the target two bunsetsus. One of the target two bunsetsus (hereafter, the forward bunsetsu) is the top bunsetsu of the shift stack. The other (hereafter, the backward bunsetsu) is the front bunsetsu of the queue if the swap stack is empty and the top bunsetsu of the swap stack otherwise. In practice, the choice of operators is limited by the state of two stacks and the queue, due to constraints on two aspects of the algorithm's behavior and Japanese grammar.
3. The repetition of 2. finishes when both the queue and the swap stack become empty and only a tree of which the root is the end-of-sentence bunsetsus is left in the shift stack.

Each operator is elaborated as follows:

Shift operator ensures that the forward bunsetsu does not depends on the backward one, and moves the backward one into the shift stack.

Shift-Comma operator performs similar to the Shift operator and also inserts a comma between the forward and backward bunsetsus.

Reduce operator specifies that the forward bunsetsu depends on the backward bunsetsu, removes the forward bunsetsu from the shift stack, and adds it as a first child node of the backward bunsetsu to form a dependency tree.

Reduce-Comma operator performs the same operation of the Reduce operator. It also inserts a comma between the forward and backward bunsetsus.

Swap operator determines to swap the order of the forward and backward bunsetsus, and pushes them into the swap stack in this order. By using the swap stack, we can reset the previous decision on dependency parsing and comma insertion related to the target two bunsetsus, and reperforms these processes based on the swapped word order again.

We describe a concrete flow of our algorithm presented in Figure 2. As can be seen in Figure 2, a box means a bunsetsu and boxes of the target two bunsetsus are shown with a bold frame. In the initial state at time 1, the bunsetsu sequence of the input sentence is stored in the queue, and both the shift stack and the swap stack are empty, so only the front of the queue is targeted, and Shift is selected. As a result, "①I" is pushed into the shift stack. At time 2, Shift is selected, and "②home" is pushed into the shift stack because it assumes that there is not a dependency relation and a comma between "①I" and "②home". At time 4, Reduce operator is chosen and "③the city" is removed from the shift stack because there is not a comma but a dependency relation between "③the city" and "④longing for". At time 5, Swap is selected, and therefore "②home", "③the city" and "④longing for" are pushed into the swap stack in the order of "②home", "④longing for", "③the city". At time 6, Shift-Comma is selected, and consequently a comma is inserted after "①I" and "③the city," which is the top of the swap stack, is pushed into the shift stack since there is no dependency relation, but then again a comma between "①I" and "③the city". Finally, at time 14, the queue and the swap stack are empty, and only the final bunsetsu is on the shift stack, so the process ends.

3.2 Probabilistic Model

In this section, we describe a probabilistic model used in the operator selection in our proposed algorithm. In this study, we conducted a probabilistic model that estimates the validity of the processing results generated by each operator, and selects each operator based on the highest value of the processing results.

In the following equation, f_t represents an operator that has been selected at time t and $\boldsymbol{f}_t = f_1 f_2 \cdots f_t$ indicates a sequence of operations from time 1 to time t. $B = b_1 b_2 \cdots b_n$ defines the bunsetsu sequence of an input sentence. b_i distinguishes the ith bunsetsu in the word order of an input sentence. $S^{\boldsymbol{f}_t}$ presents the structure expressing the result that $\boldsymbol{f}_t$ (operations up to time t) are performed. The structure $S^{\boldsymbol{f}_t}$ is defined as a tuple $S^{\boldsymbol{f}_t} = \langle O^{\boldsymbol{f}_t}, C^{\boldsymbol{f}_t}, D^{\boldsymbol{f}_t} \rangle$, where $O^{\boldsymbol{f}_t} = \{o^{\boldsymbol{f}_t}_{1,2}, o^{\boldsymbol{f}_t}_{2,3}, \cdots, o^{\boldsymbol{f}_t}_{1,n}, o^{\boldsymbol{f}_t}_{2,3}, \cdots, o^{\boldsymbol{f}_t}_{i,j}, \cdots, o^{\boldsymbol{f}_t}_{n-1,n}\}$, $C^{\boldsymbol{f}_t} = \{c^{\boldsymbol{f}_t}_{1,2}, c^{\boldsymbol{f}_t}_{2,3}, \cdots, c^{\boldsymbol{f}_t}_{1,n}, c^{\boldsymbol{f}_t}_{2,3}, \cdots, c^{\boldsymbol{f}_t}_{i,j}, \cdots, c^{\boldsymbol{f}_t}_{n-1,n}\}$, and $D^{\boldsymbol{f}_t} = \{d^{\boldsymbol{f}_t}_{1,2}, d^{\boldsymbol{f}_t}_{2,3}, \cdots, d^{\boldsymbol{f}_t}_{1,n}, d^{\boldsymbol{f}_t}_{2,3}, \cdots, d^{\boldsymbol{f}_t}_{i,j}, \cdots, d^{\boldsymbol{f}_t}_{n-1,n}\}$ are the word order, the comma positions, and the dependency structure, respectively, which were determined by $\boldsymbol{f}^t$.

Here, $o^{\boldsymbol{f}_t}_{i,j} (1 \leq i < j \leq n)$ expresses the order between b_i and b_j after an operation at time t, and $o^{\boldsymbol{f}_t}_{i,j}$ is 1 if b_i is located before b_j after a operation $\boldsymbol{f}_t$, and is 0 otherwise. In addition, $c^{\boldsymbol{f}_t}_{i,j} (1 \leq i < j \leq n)$ is 1 if there is a comma between b_i and b_j, and is 0 otherwise. Fi-

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Table 1: Experimental Results.

word reordering		comma insertion		
pair	complete	recall	precision	f-measure
72.61% (24,209/33,343)	10.90% (109/1,000)	48.44% (31/64)	60.78% (31/51)	53.91

nally, $d_{i,j}^{f_t}(1 \leq i < j \leq n)$ expresses the dependency relation between b_i and b_j after an operation at time t, and $d_{i,j}^{f_t}$ is 1 if b_i depends on b_j , and is 0 otherwise.

In the proposed method, the selection of the operation f_t is calculated by Equation (1). The forward bunsetsu at time t is denoted by b_i and the backward bunsetsu is identified by b_j, and the difference between $S^{f_{t-1}f_t}$ and $S^{f_{t-1}}$ is $o_{i,j}^{f_{t-1}f_t}, c_{i,j}^{f_{t-1}f_t}, d_{i,j}^{f_{t-1}f_t}$.

$$\begin{aligned}
&argmax_{f_t} P(S^{f_{t-1}f_t}|B, S^{f_{t-1}}) \\
&\approx argmax_{f_t} P(o_{i,j}^{f_{t-1}f_t}, c_{i,j}^{f_{t-1}f_t}, d_{i,j}^{f_{t-1}f_t}|B, S^{f_{t-1}}) \\
&= argmax_{f_t} \Big(P(o_{i,j}^{f_{t-1}f_t}|B, S^{f_{t-1}}) \\
&\times P(c_{i,j}^{f_{t-1}f_t}|B, S^{f_{t-1}}, o_{i,j}^{f_{t-1}f_t}) \\
&\times P(d_{i,j}^{f_{t-1}f_t}|B, S^{f_{t-1}}, o_{i,j}^{f_{t-1}f_t}, c_{i,j}^{f_{t-1}f_t}) \Big)
\end{aligned} \tag{1}$$

$P(o_{i,j}^{f_{t-1}f_t}|B, S^{f_{t-1}})$, $P(c_{i,j}^{f_{t-1}}|B, S^{f_{t-1}}, o_{i,j}^{f_{t-1}f_t})$ and $P(d^{f_{t-1}}|B, S^{f_{t-1}}, o_{i,j}^{f_{t-1}f_t}, c_{i,j}^{f_{t-1}f_t})$ are all estimated by machine learning. In estimating $P(o_{i,j}^{f_{t-1}f_t}|B, S^{f_{t-1}})$, we used all of the features proposed by (Yoshida et al., 2014) except for the features related to the bunsetsu that are depended to the forward bunsetsu or backward one. In estimating $P(c_{i,j}^{f_{t-1}f_t}|B, S^{f_{t-1}}, o_{i,j}^{f_{t-1}f_t})$, we applied all of the features suggested by (Murata et al., 2010) that can be acquired without the dependency information about the forward bunsetsu. In estimating $P(d_{i,j}^{f_{t-1}f_t}|B, S^{f_{t-1}}, o_{i,j}^{f_{t-1}f_t}, c_{i,j}^{f_{t-1}f_t})$, all of the features recommended by (Sassano, 2004) were confidered.

4 EXPERIMENT

To evaluate the performance of the proposed method for word reordering and comma insertion in hard-to-read Japanese sentences, we conducted an experiment using newspaper articles.

4.1 Creation of Test Data

According to the assumption that sentences in newspaper articles are written in easy-to-read word order, we created 1000 sentences with hard-to-read word order based on the following procedure: (1) Create pseudo-sentences with hard-to-read word order from a newspaper article (Yoshida et al., 2014). (2) Manually add commas to make it as easy as possible to read in that word order.

4.2 Outline of Experiment

We employed a gradient-boosting machine (GBM) as the machine learning model for estimating each probability in Equation (1) and used the default LightGBM[2]. A total of 35,404 sentences from Kyoto University Text Corpus Ver. 4.0 (Kawahara et al., 2002) were selected for the training, excluding the sentences used to create the test data for Section 4.1.

In the evaluation of word reordering, we obtained the following two measurements, which are defined by (Uchimoto et al., 2000). **Complete agreement** is the percentage of the output sentences in which the word order entirely agrees with that of the original sentence. **Pair agreement** is the percentage of the pairs of bunsetsus whose word order agrees with the word order in the original sentence.

For the comma insertion, we evaluated only sentences of which the word order completely agrees. We measured the precision and recall based on the assumption that the comma positions in Kyoto University Text Corpus are correct, as with (Murata et al., 2010).

4.3 Experimental Results

Table 1 presents the pair and complete agreements of our method for word reordering. It also shows the recall, precision, and F-measure for the comma insertion in the output sentences whose word order is completely consistent with the correct answer. Although the performance of word reordering and comma insertion was not necessarily sufficient, there existed 65 sentences in which the word order and comma positions matched perfectly between the output sentences and the correct answers as shown in Figure 3. Therefore, it can conclude that the feasibility of our method was confirmed.

[2]https://lightgbm.readthedocs.io/en/latest/

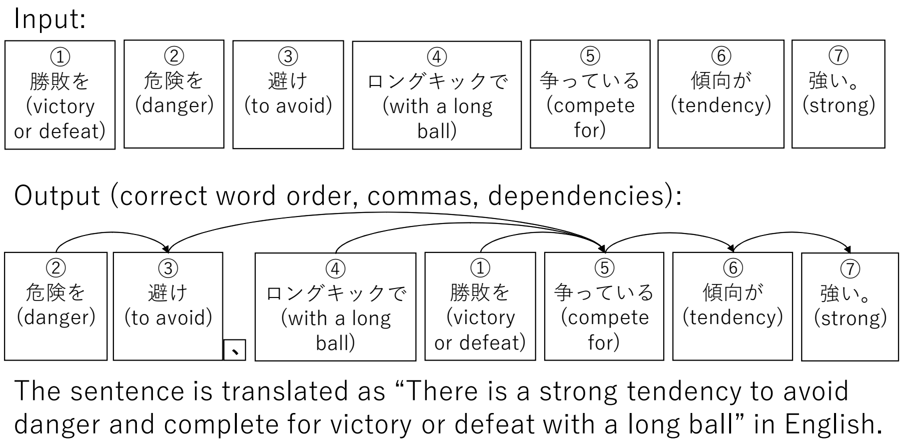

Figure 3: Successful example of word ordering, comma insertion, and dependency parsing.

5 CONCLUSION

This paper proposed a method of simultaneously determining appropriate word order, appropriate comma positions, and the dependency structure for an input sentence which has unsuitable word sequence. The proposed algorithm in this study is further extended the Shift-Reduce algorithm suggested (Sassano, 2004) and focused on two local bunsetsus. We confirmed the feasibility of our method as the results of the evaluation experiment using 1000 sentences with hard-to-read word order.

For future works, we are interested in improving the agreements on word reordering, and the precision and recall on comma insertion by reviewing the features and the machine learning method used for probability estimation.

ACKNOWLEDGMENTS

This work was partially supported by JSPS KAKENHI Grant Numbers JP26280082, JP16K00300, JP19K12127.

REFERENCES

Belz, A. and Kow, E. (2011). Discrete vs. continuous rating scales for language evaluation in NLP. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 230–235.

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

Harbusch, K., Kempen, G., van Breugel, C., and Koch, U. (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*, pages 9–11.

Kawahara, D., Kurohashi, S., and Hasida, K. (2002). Construction of a Japanese relevance-tagged corpus. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pages 2008–2013.

Kruijff, G.-J. M., Kruijff-Korbayová, I., Bateman, J., and Teich, E. (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*, pages 74–83.

Murata, M., Ohno, T., and Matsubara, S. (2010). Automatic comma insertion for Japanese text generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 892–901.

Ringger, E., Gamon, M., Moore, R. C., Rojas, D., Smets, M., and Corston-Oliver, S. (2004). Linguistically informed statistical models of constituent structure for ordering in sentence realization. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 673–679.

Sassano, M. (2004). Linear-time dependency analysis for Japanese. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 8–14.

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

Uchimoto, K., Murata, M., Ma, Q., Sekine, S., and Isahara, H. (2000). Word order acquisition from corpora. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 871–877.

Yokobayashi, H., Suganuma, A., and Taniguchi, R.-i. (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).

Yoshida, K., Ohno, T., Kato, Y., and Matsubara, S. (2014). Japanese word reordering integrated with dependency parsing. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1186–1196.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS