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

Kota Miyachi¹, Tomohiro Ohno², and Shigeki Matsubara³

¹Graduate School of Informatics, Nagoya University, Nagoya, Japan
²Graduate School of Advanced Science and Technology, Tokyo Denki University, Tokyo, Japan
³Information & Communications, Nagoya University, Nagoya, Japan

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

Abstract: Japanese has widely been recognized as a 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 a 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-
33

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. Besides, 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 insertion.

1Bunsetsu 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.
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.
tion in our method. Our method processes the bunsetsu sequence of an input sentence in order from the beginning 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 bunsetsu is left in the shift stack.

Each operator is elaborated as follows:

**Shift** operator ensures that the forward bunsetsu does not depend 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. 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, “⒃I” is pushed into the shift stack. At time 2, Shift is selected, and “⃼the city” is pushed into the shift stack because it assumes that there is not a dependency relation and a comma between “⃼I” and “⃼the city”. At time 4, Reduce operator is chosen and “⃼the city” is removed from the shift stack because there is not a comma but a dependency relation between “⃼the city” and “⃼the city”. As can be seen in Figure 2, a box presents a bunsetsu and boxes of the target two bunsetsus, and reperforms the operator that has been selected at time t. B = b₁b₂…bₙ defines the bunsetsu sequence of an input sentence, bₗ distinguishes the lth bunsetsu in the word order of an input sentence. Sₗ presents the structure expressing the result that f₁ (operations up to time t) are performed. The structure Sₙ is defined as a tuple Sₙ = (Oₙ,cₙ,Dₙ), where Oₙ = {o₁₂, o₂₃, …, oₙ₋₁ₙ}, d₁₂, d₂₃, …, dₙ₋₁ₙ}, and Dₙ = {d₁₂, d₂₃, …, dₙ₋₁ₙ} are the word order, the comma positions, and the dependency structure, respectively, which were determined by fₙ.

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.

3.2 Probabilistic Model

In the following equation, f₁ represents an operator that has been selected at time t and f₁ = f₁2…fₙ indicates a sequence of operations from time 1 to time t. B = b₁b₂…bₙ defines the bunsetsu sequence of an input sentence, bₗ distinguishes the lth bunsetsu in the word order of an input sentence. Sₗ presents the structure expressing the result that f₁ (operations up to time t) are performed. The structure Sₙ is defined as a tuple Sₙ = (Oₙ,cₙ,Dₙ), where Oₙ = {o₁₂, o₂₃, …, oₙ₋₁ₙ}, d₁₂, d₂₃, …, dₙ₋₁ₙ}, and Dₙ = {d₁₂, d₂₃, …, dₙ₋₁ₙ} are the word order, the comma positions, and the dependency structure, respectively, which were determined by fₙ.

Here, o₁₂(1 ≤ i < j ≤ n) expresses the order between bᵢ and bⱼ after an operation at time t, and o₁₂ is 1 if bᵢ is located before bⱼ after a operation fᵢ, and is 0 otherwise. In addition, c₁₂(1 ≤ i < j ≤ n) is 1 if there is a comma between bᵢ and bⱼ, and is 0 otherwise. Fi-
Table 1: Experimental Results.

<table>
<thead>
<tr>
<th>word reordering</th>
<th>comma insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>pair</td>
<td>recall</td>
</tr>
<tr>
<td>complete</td>
<td>72.61%</td>
</tr>
<tr>
<td>(24,209/33,343)</td>
<td>(109/1,000)</td>
</tr>
</tbody>
</table>

nally, \(d_{ij}^f(1 \leq i < j \leq n)\) expresses the dependency relation between \(b_i\) and \(b_j\) after an operation at time \(t\), and \(d_{ij}^b\) 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 \(b_t\) and the backward bunsetsu is identified by \(b_j\), and the difference between \(S_f\) and \(S_{f-1}\) is \(o_{ij}^{f-1}, f_{ij}^{f-1}, d_{ij}^{f-1}\).

\[
\begin{align*}
\text{argmax}_f & \ P(S_f|B, S_{f-1}) \\
& \approx \text{argmax}_f P(o_{ij}^{f-1}, f_{ij}^{f-1}, d_{ij}^{f-1}|B, S_{f-1}) \\
& = \text{argmax}_f \left( P(o_{ij}^{f-1}|B, S_{f-1}) \right) \\
& \times P(f_{ij}^{f-1}|B, S_{f-1}, o_{ij}^{f-1}) \\
& \times P(d_{ij}^{f-1}|B, S_{f-1}, f_{ij}^{f-1})
\end{align*}
\]

\(P(o_{ij}^{f-1}|B, S_{f-1})\), \(P(f_{ij}^{f-1}|B, S_{f-1}, o_{ij}^{f-1})\) and \(P(d_{ij}^{f-1}|B, S_{f-1}, f_{ij}^{f-1})\) are all estimated by machine learning. In estimating \(P(o_{ij}^{f-1}|B, S_{f-1})\), we used all of the features proposed by Yoshida et al. (2014) except for the features related to the bunsetsu that are dependent to the forward bunsetsu or backward one. In estimating \(P(f_{ij}^{f-1}|B, S_{f-1}, o_{ij}^{f-1})\), 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 \(P(d_{ij}^{f-1}|B, S_{f-1}, f_{ij}^{f-1})\), all of the features recommended by Sassano (2004) were considered.

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 bunsetsu 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/
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


Yokoyabashi, H., Suganuma, A., and Taniguchi, R.-i. (2004). Generating candidates for rewriting based on...