Dependency Structure for Incremental Parsing of Japanese and Its Application

Tomohiro Ohno
Information Technology Center,
Nagoya University
Nagoya, Japan
ohno@nagoya-u.jp

Shigeki Matsubara
Graduate Schoool of Information Science,
Nagoya University
Nagoya, Japan
matubara@nagoya-u.jp

Abstract

Previous researches on incremental dependency parsing have not discussed how a parser should output the information about dependency relation where the modified word has not been inputted yet. This paper proposes a dependency structure which a Japanese incremental dependency parser should produce in the middle of an input sentence. The dependency structure also expresses the fact that a word depends on a word which the parser would receive later. In addition, we present a method for incremental dependency parsing to produce our proposed dependency structure. As the result of an experiment on incremental parsing, we confirmed the feasibility of our incremental dependency parsing. Furthermore, this paper shows its application in the simultaneous linefeed insertion for real-time generation of more readable captions, and then describes the effectiveness of our proposed dependency structure.

1 Introduction

In applications such as simultaneous machine interpretation, spoken dialogue and real-time captioning, it is necessary to execute processing simultaneously with speech input, and thus, the questions of what syntactic information is acquirable and when the information can be acquired become issues. However, previous researches on incremental dependency parsing (Kato et al., 2005; Johansson and Nugues, 2007; Nivre, 2008) have not discussed how a parser should output the information about dependency relation where the modified word has not been inputted.

This paper proposes a dependency structure which a Japanese incremental dependency parser should produce in the middle of an input sentence. Our proposed dependency structure makes a parser clarify the fact that the modified *bunsetsu*[1] of a bunsetsu will be inputted later. This enables a higher layer application to use the information that the bunsetsu does not depend on any bunsetsus which have been already inputted. Especially, in the case of Japanese, since the modified bunsetsu is always inputted after the modifier bunsetsu, our proposal is effective for the applications which execute processing simultaneously with speech input.

In addition, this paper describes a method for incremental dependency parsing to produce our proposed dependency structure whenever receiving a bunsetsu. We conducted an experiment on incremental parsing and confirmed the feasibility of our method. Furthermore, this paper shows its application to the simultaneous linefeed insertion for real-time generation of more readable captions. As the result of an experiment on linefeed insertion, we confirmed the effectiveness of our proposed dependency structure.

2 Dependency structure for incremental parsing

Since applications such as simultaneous machine interpretation and real-time captioning need to execute the process simultaneously with speech input, it is desirable for them to be able to acquire the dependency information at any time. In our research, incremental dependency parsing shall output the parsing result whenever receiving a bunsetsu.

[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.

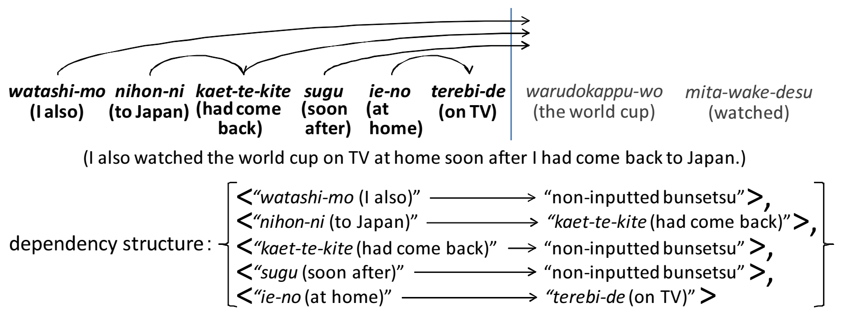
watashi-mo (I also)
nihon-ni (to Japan)
kaet-te-kite (had come back)
sugu (soon after)
ie-no (at home)
terebi-de (on TV)
warudokappu-wo (the world cup)
mita-wake-desu (watched)
(I also watched the world cup on TV at home soon after I had come back to Japan.)
dependency structure :
<"watashi-mo (I also)" → "non-inputted bunsetsu">,
<"nihon-ni (to Japan)" → "kaet-te-kite (had come back)">,
<"kaet-te-kite (had come back)" → "non-inputted bunsetsu">,
<"sugu (soon after)" → "non-inputted bunsetsu">,
<"ie-no (at home)" → "terebi-de (on TV)">

Figure 1: Example of a dependency structure outputted in the middle of an input sentence

Next, we discuss the output contents of incremental dependency parsing. When applications use incremental dependency parsing, the information about whether or not the dependency structure of a sequence of busnetsus is closed, that is, the information about a syntactically sufficient unit is also useful. In fact, when identifying the timing of the start of interpreting in the simultaneous machine interpretation (Ryu et al., 2006; Fügen et al., 2007; Paulik and Waibel, 2010), the timing of inserting back-channel feedbacks (Fujie et al., 2004; Kamiya et al., 2010), and the appropriate position of a linefeed in captioning (Ohno et al., 2009), the information about a syntactically sufficient unit is used as an important clue.

In our research, an incremental dependency parsing shall clearly output the fact that a bunsetsu depends on a bunsetsu which will be later inputted. If it becomes clear that the modified bunsetsu of a bunsetsu has not been inputted yet, the higher layer applications can identify the syntactically sufficient units in the inputted sequence of bunsetsus and use the information effectively.

In what follows, we describe the dependency structure which incremental dependency parsing in our research outputs in the middle of an input sentence. When a sentence S composed of a sequence of bunsetsus $b_1 \cdots b_n$ is parsed[2], we define the dependency structure D_x which is outputted at the time that a bunsetsu $b_x (1 \leq x \leq n)$ is inputted as follows:

$$D_x = \left\{ \langle b_1 \rightarrow dep(b_1) \rangle, \cdots, \langle b_{x-1} \rightarrow dep(b_{x-1}) \rangle \right\}$$

$$\left(\text{Here, } dep(b_y) \in \left\{ \begin{array}{ll} \{b_{y+1}, \cdots, b_x, b_{over}\} & (\text{if } b_x \neq b_n) \\ \{b_{y+1}, \cdots, b_x\} & (\text{otherwise}) \end{array} \right\} \atop (1 \leq y \leq x-1) \right)$$

[2]In this research, we assume that sentence boundaries are detected as its preprocessing.

where $\langle b_y \rightarrow dep(b_y) \rangle$ means the dependency relation where the modifier bunsetsu is b_y and where the modified bunsetsu is $dep(b_y)$. $dep(b_y)$ is a member of a set which contains not only $b_{y+1}, \cdots, b_x$, which are inputted after the bunsetsu b_y, but also b_{over}. $dep(b_y) = b_{over}$ means that the modified bunsetsu of b_y is not any of $b_{y+1}, \cdots, b_x$ but a bunsetsu which will be inputted later. In addition, we assume the following three Japanese syntactic constraints:

- No dependency is directed from right to left.
- Dependencies don't cross each other.
- Each bunsetsu, except the final one in a sentence, depends on only one bunsetsu.

Figure 1 shows an example of a dependency structure outputted in the middle of an input sentence. This example shows the dependency structure which is outputted right after the bunsetsu "*terebi-de* (on TV)" was inputted when a parser parses the sentence "*watashi-mo nihon-ni kaet-te-kite sugu ie-no terebi-de warudokappu-wo mita-wake-desu* (I also watched the world cup on TV at home soon after I had come back to Japan.)"

3 Incremental dependency parsing

This section describes a method to produce the dependency structure proposed in Section 2 whenever receiving a bunsetsu. This method is implemented by improving the dependency parsing method proposed by Uchimoto et al. (2000).

Generally, stochastic dependency parsing methods identify the dependency structure which maximizes the conditional probability $P(D|S)$, which means the probability that the dependency structure for a sentence S is D. In usual methods, a relationship between two bunsetsus is tagged with a "1" or "0" to indicate whether or not there is a dependency relation between the bunsetsus, respectively. On the other hand, in the Uchimoto et al.'s method, a relationship between two bunsetsus is tagged with a "between," "depend," or "beyond" to indicate the following three cases, respectively. The anterior bunsetsu can depend on (1) a bunsetsu between the two, (2) the posterior bunsetsu, or (3) a bunsetsu beyond the posterior one. Then, the dependency probability of two bunsetsus is estimated by using the product of the probabilities of the relationship between the left bunsetsu and those to its right in a sentence.

In our research, we thought that by adapting the basic idea in the Uchimoto et al.'s method, we might be able to calculate the probability that a bunsetsu does not depend on any bunsetsus which have been already inputted, that is, the probability that a bunsetsu depends on a non-inputted bunsetsu (b_{over}). As a difference from the Uchimoto et al.'s method, our method identifies the dependency structure D_x which should be outputted when the sequence of bunsetsus $B_x = b_1, \cdots, b_x (1 \leq x \leq n)$, which is a subsequence of a sentence $S = b_1 \cdots b_n$, is inputted. Our method outputs the dependency structure which maximizes the probability $P(D_x|B_x)$ as the most appropriate dependency structure for the sequence of bunsetsus B_x.

Here, D_x is described as an ordered set $\{d_1, \cdots, d_{x-1}\}$ of dependency relations d_i $(1 \leq i \leq x-1)$, where the modifier bunsetsu is a bunsetsu b_i. In addition, d_i is described as $d_i = \{d_{i,i+1}, \cdots, d_{i,x}, d_{i,x+1}\}$ $(1 \leq i \leq x-1)$. $d_{i,i+j}$ is a flag which indicates the relationship between bunsetsus b_i and b_{i+j}, and takes the following value:

$$d_{i,i+j} = \left\{ \begin{array}{ll} 0 & (1 \leq j < dist(i)) \\ 1 & (j = dist(i)) \\ 2 & (dist(i) < j \leq n-i) \end{array} \right\}$$

where $dist(i)$ is defined as $dist(i) = l - i$ when a bunsetsu b_i depends on a bunsetsu $b_l (i < l \leq x+1 \leq n)$. Note that our method adds a flag $d_{i,x+1}$, which indicates the relationship between b_i and a non-inputted bunsetsu b_{x+1}, to the elements of d_i as a dummy.

Using the above-mentioned notation, our method calculates $P(D_x|B_x)$ as follows[3]:

$$P(D_x|B_x)^2 = \prod_{i=1}^{x-1} P(d_i|B_x)$$

$$= \prod_{i=1}^{x-1} P(d_{i,i+1}, \cdots, d_{i,x+1}|B_x)$$

$$= \prod_{i=1}^{x-1} \left(\prod_{j=1}^{dist(i)-1} P(d_{i,i+j} = 0|B_x) \right. \\ \times P(d_{i,i+dist(i)} = 1|B_x) \\ \left. \times \prod_{j=dist(i)+1}^{x-i+1} P(d_{i,i+j} = 2|B_x) \right)$$

[3] Uchimoto et al. (2000a) discussed the rationale and assumptions behind the form of this model.

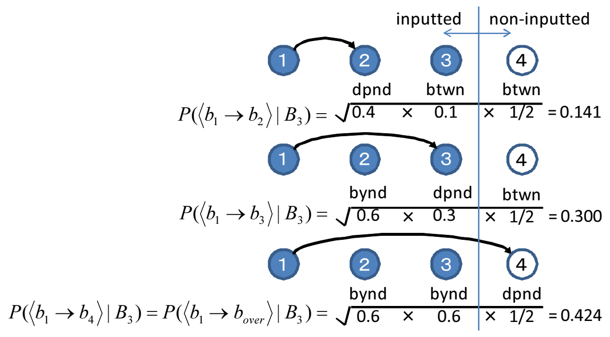
inputted
non-inputted
1
2
3
4
dpnd
btwn
btwn
$P(\langle b_1 \to b_2 \rangle | B_3) = \sqrt{0.4 \times 0.1 \times 1/2} = 0.141$
bynd
dpnd
btwn
$P(\langle b_1 \to b_3 \rangle | B_3) = \sqrt{0.6 \times 0.3 \times 1/2} = 0.300$
bynd
bynd
dpnd
$P(\langle b_1 \to b_4 \rangle | B_3) = P(\langle b_1 \to b_{over} \rangle | B_3) = \sqrt{0.6 \times 0.6 \times 1/2} = 0.424$

Figure 2: Calculation of dependency probability

where the value of $dist(i)$ is decided based on the D_x. Here, we use the following value as $P(d_{i,i+j}|B_x)$, which means the probability of the relationship ("beyond," "depend," or "between") between two bunsetsus.

- When $1 \leq j \leq x - i$: $P(d_{i,i+j}|B_x) =$ the value which is estimated using the maximum entropy method in the same way[4] as the Uchimoto et al.'s method.
- When $j = x - i + 1$: $P(d_{i,i+j} = 0|B_x) = 0$, $P(d_{i,i+j} = 1|B_x) = P(d_{i,i+j} = 2|B_x) = 1/2$

The reasons that we decided the second itemization are as follows: The relationship $d_{i,x+1}$ between a bunsetsu b_i and a non-inputted bunsetsu b_{x+1} does not become the relationship "beyond," but becomes either "depend" or "between." Moreover, we assume that the probability that the relationship becomes "depend" is equal to that of "between" because the probabilities are unknown. This assumption enables $argmax_{D_x} P(D_x|B_x)$ to be calculated with no relation to the probabilities between b_i and b_{x+1}.

Finally, if there exists a $\langle b_i \to b_{x+1} \rangle$ in the dependency structure D_x which maximizes $P(D_x|B_x)$,

[4] However, the features which we used in the maximum entropy method are the same as those used by Ohno et al. (2007).

our method outputs the dependency structure which is made by replacing every $\langle b_i \rightarrow b_{x+1}\rangle$ in the D_x with $\langle b_i \rightarrow b_{over}\rangle$.

Figure 2 shows how to calculate the dependency probability $P(d_1|B_3)$ when a bunsetsu b_3 in a sentence composed of $b_1, \cdots, b_5$ is inputted. Note that we consider $P(\langle b1 \rightarrow b_4\rangle|B_3) = P(\langle b1 \rightarrow b_{over}\rangle|B_3)$.

4 Experiment

To evaluate the feasibility of incrementally producing the dependency structure proposed in Section 2, we conducted an experiment on dependency parsing.

4.1 Outline of experiment

As the experimental data, we used the transcribed data of Japanese discourse speech in the Simultaneous Interpretation Database (Matsubara et al., 2002). All the data had been manually annotated with information on morphological analysis, bunsetsu boundary detection, clause boundary detection and dependency analysis (Ohno et al., 2009).

We performed a cross-validation experiment by using 16 discourses. That is, we repeated the experiment, in which we used one discourse from among 16 discourses as the test data and the others as the learning data, 16 times. However, since we used 2 discourses among 16 discourses as the preliminary analysis data, we evaluated the experimental results for the remaining 14 discourses (1,714 sentences, 20,707 bunsetsus). Here, as the morphological information, bunsetsu boundary information, and clause boundary information in the input of dependency parsing, we used one which had been manually annotated. In addition, we used the maximum entropy method tool (Zhang, 2013) with the default options except "-i (iteration) 2000."

4.2 Evaluation index

Our method is designed to output the dependency structure defined in Section 2 whenever a bunsetsu is inputted. Therefore, we introduced the following new evaluation index to evaluate the accuracy of incremental dependency parsing:

$$acc. = \frac{\sum_i^N \sum_{j=1}^{n_i} DepNum(Match(D_j^i, G_j^i))}{\sum_i^N \sum_{j=1}^{n_i} DepNum(D_j^i)}$$

where D_j^i and G_j^i indicate the dependency structure which a parser outputs and the correct one which

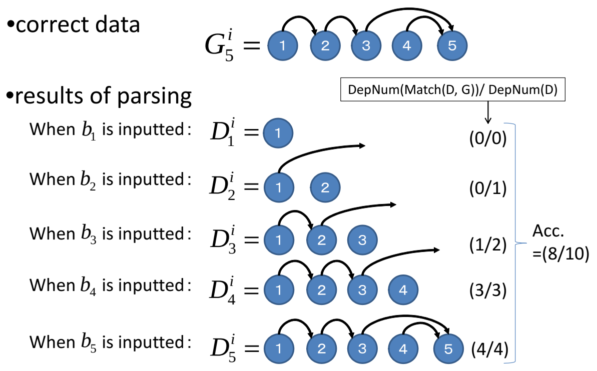
•correct data
$G_5^i =$
1
2
3
4
5
•results of parsing
DepNum(Match(D, G))/ DepNum(D)
When b_1 is inputted : $D_1^i =$
When b_2 is inputted : $D_2^i =$
When b_3 is inputted : $D_3^i =$
When b_4 is inputted : $D_4^i =$
When b_5 is inputted : $D_5^i =$
(0/0)
(0/1)
(1/2)
(3/3)
(4/4)
Acc. =(8/10)

Figure 3: Calculation of accuracy

	recall	precision	f-measure
non-inputted	70.4% (22,811/ 32,389)	75.0% (22,811/ 30,413)	72.6%
inputted	74.8% (102,778/ 137,376)	73.8% (102,778/ 139,352)	74.3%

Table 1: Recalls and precisions of our method

a parser should output respectively when a bunsetsu $b_j (1 \leq j \leq n_i)$ in a sentence $S_i (1 \leq i \leq N)$ is inputted. In addition, $DepNum()$ is a function which returns the number of elements of the input set of dependency relations, and $Match()$ is a function which returns an intersection between the two input sets of dependency relations.

Figure 3 shows an example of the calculation of our defined accuracy in case of parsing a sentence S_i composed of 5 bunsetsus. Here, we explain how to calculate the $DepNum(Match(D_3^i, G_3^i))$ when the 3rd bunsetsu is inputted. From the correct data of the dependency structure of a whole sentence S_i, $G_3^i = \{\langle b_1 \rightarrow b_2\rangle, \langle b_2 \rightarrow b_3\rangle\}$ is derived. On the other hand, the result of parsing is $D_3^i = \{\langle b_1 \rightarrow b_2\rangle, \langle b_2 \rightarrow b_{over}\rangle\}$. From the G_3^i and D_3^i, $DepNum(Match(D_3^i, G_3^i)) = 1$ is derived.

Furthermore, we obtained the recalls and precisions, separately for the case that the modified bunsetsu has not been inputted and the case that the modified bunsetsu has been inputted. The recall and precision for the former case are respectively defined as follows:

$$rec. = \frac{\sum_i^N \sum_{j=1}^{n_i} DepNum(Over(Match(D_j^i, G_j^i)))}{\sum_i^N \sum_{j=1}^{n_i} DepNum(Over(G_j^i))}$$

$$prec. = \frac{\sum_i^N \sum_{j=1}^{n_i} DepNum(Over(Match(D_j^i, G_j^i)))}{\sum_i^N \sum_{j=1}^{n_i} DepNum(Over(D_j^i))}$$

where Over() is a function which returns a set of dependency relations where the modified bunsetsu has not been inputted yet from among the input set of dependency relations. The recall and precision for the latter case are defined by replacing the Over() in the above-mentioned definition with the NonOver(). NonOver() is a function which returns a set of dependency relations where the modified bunsetsu has been inputted from among the input set of dependency relations.

4.3 Experimental results

The accuracy obtained using our method was 74.0% (125,589/169,765). Only as a guide, we conducted a parsing experiment by using the Uchimoto et al.'s method[5] and CaboCha(Kudo and Matsumoto, 2002), which identify the dependency structure after the whole sentence is inputted. As the result of measuring the conventional dependency accuracy (= $\sum_i^N DepNum(Match(D_{n_i}^i, G_{n_i}^i)) / \sum_i^N DepNum(D_{n_i}^i)$), the dependency accuracy of the Uchimoto et al.'s method and CaboCha were 75.8% (14.391/18,993) and 78.0% (14,814/18,993) respectively.

Table 1 shows the recalls and precisions of our method, separately for the case that the modified bunsetsu has not been inputted and the case that the modified bunsetsu has been inputted. The f-measure for the case that the modified bunsetsu has not been inputted was not as high as that for the non-inputted case. However, the precision for the inputted case was higher than that for the non-inputted case. We confirmed the feasibility of outputting our proposed dependency structure whenever a bunsetsu is inputted.

5 Application to sequential linefeed insertion

In this section, we describe application of our incremental dependency parsing described in Section 3 to sequential linefeed insertion.

5.1 Sequential linefeed insertion

Whenever a bunsetu is inputted, our method decisively judges whether or not a linefeed should be inserted between the inputted bunsetsu and the adjacent

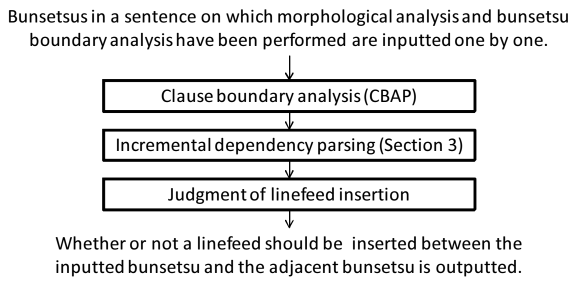
Bunsetsus in a sentence on which morphological analysis and bunsetsu boundary analysis have been performed are inputted one by one.
Clause boundary analysis (CBAP)
Incremental dependency parsing (Section 3)
Judgment of linefeed insertion
Whether or not a linefeed should be inserted between the inputted bunsetsu and the adjacent bunsetsu is outputted.

Figure 4: Flow of sequential linefeed insertion

bunsetsu. Here, morphological analysis and bunsetsu boundary analysis have been already performed for the input sequence of bunsetsus in a sentence.

Figure 4 shows the flow of our sequential linefeed insertion. First, our method decides whether there exists a clause boundary between the inputted bunsetsu b_x and the adjacent bunsetsu b_{x-1} or not by using Clause Boundary Annotation Program (CBAP) (Kashioka and Maruyama, 2004). Next, our method parses the dependency structure D_x proposed in Section 2 by using the incremental dependency parsing method proposed in Section 3.

Finally, in judgment of linefeed insertion, our method decisively judges whether a linefeed should be inserted between the inputted bunsetsu b_x and the adjacent bunsetsu b_{x-1} by using the maximum entropy method. The maximum entropy method estimates the probability that a linefeed should be inserted at a point by considering such information as morphological analysis results, bunsetsu boundaries, clause boundaries, and dependency structures. When the probability is larger than 0.5, our method decides to insert a linefeed at the point.

The features used in the ME method, which estimates the probability that a linefeed is inserted between a bunsetsu b_{x-1} and b_x, are as roughly the same as those for the conventional sentence-based method (Ohno et al., 2009)[6]. The difference is that our method does not use the following three features among those for the sentence-based method:

- whether or not b_{x-1} depends on the final bunsetsu of a clause
- whether or not b_{x-1} depends on a bunsetsu to

[5]We used the same features in Section 3 when estimating the probability of the relationship between two bunsetsus.

[6]The sentence-based method decides the most appropriate linefeed insertion points in the whole of a sentence after the sentence-end bunsetsu is inputted.

	recall	precision	f-measure
our method	79.6% (4,375/5,497)	70.2% (4,375/6,228)	74.6%
conv. method	73.7% (4,052/5,497)	74.4% (4,052/5,447)	74.0%

Table 2: Experimental results of linefeed insertion

which the number of characters from the start of the line is less than or equal to the maximum number of characters (20 characters)

- whether or not there exists a bunsetsu which depends on the modified bunsetsu of b_{x-1}, among bunsetsus which are located after b_{x-1} and to which the number of characters from the start of the line is less than or equal to the maximum number of characters

The values of these features can not be decided until the modified bunsetsu of b_{x-1} is inputted if b_{x-1} does not depend on b_x. Since our sequential linefeed insertion method needs to identify the linefeed insertion points simultaneously with speech input, our method does not use these features.

5.2 Experiment on linefeed insertion

To evaluate the application potency of the dependency structure proposed in Section 2, we conducted an experiment on linefeed insertion using the sequential linefeed insertion method described in the previous section.

5.2.1 Experimental outline

We used the same experimental data and performed a cross-validation experiment in the same way as Section 4.1. However, when inputting the test data, we eliminated the information on clause boundaries, dependency, and linefeeds. In the evaluation, we obtained the recall, precision and f-measure on the linefeed insertion performance.

For comparison, we also performed linefeed insertion using the conventional sentence-based method (Ohno et al., 2009). Here, when performing the conventional method, clause boundary information and dependency information were beforehand provided using Clause Boundary Annotation Program (CBAP) and Uchimoto et al.'s dependency parsing method (Uchimoto et al., 2000b), respectively.

5.2.2 Experimental results on linefeed insertion

Table 2 shows the recalls, precisions and f-measures of our method and conventional method. The f-measure for our method was 74.6%, which was slightly larger than that for the conventional method. We confirmed that our method can output linefeed insertion points simultaneously with speech input, maintaining the approximately-same f-measure of linefeed insertion as that for the conventional method.

However, we measured the sentence accuracy, which is the ratio of sentences of which all linefeed insertion points were correctly detected. The sentence accuracies of our method and the conventional method were 35.8% (614/1,714) and 46.2% (792/1,714) respectively. The conventional method decides the most appropriate linefeed insertion points by considering the whole sentence while our method judges whether a linefeed should be inserted or not whenever a bunsetsu is inputted. This difference is thought to have caused the result. We will confirm that the readability of captions generated by our method does not decrease, by conducting not only objective evaluation based on comparing the linefeed insertion result with the correct data but also subjective evaluation.

6 Conclusion

This paper proposed the dependency structure which an incremental dependency parser outputs for an inputted bunsetsu sequence. Our proposed dependency structure makes a parser clarify the fact that the modified bunsetsu is a bunsetsu which will be inputted later. In addition, we proposed a method for incremental dependency parsing to output our proposed dependency structure. As the result of an experiment on incremental parsing, we confirmed the feasibility of our method. Furthermore, we applied our proposed incremental dependency parsing to the simultaneous linefeed insertion, and then we confirmed the application potency of our proposed dependency structure.

Acknowledgments

This research was partially supported by the Grant-in-Aid for Young Scientists (B) (No.25730134) and Scientific Research (B) (No.22300051) of JSPS.

References

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine Translation*, 21:209–252. 10.1007/s10590-008-9047-0.

Shinya Fujie, Kenta Fukushima, and Tetsunori Kobayashi. 2004. A conversation robot with back-channel feedback function based on linguistic and nonlinguistic information. In *Procedings of the 2nd International Conference on Autonomous Robots and Agents (ICARA2004)*, pages 379–384.

Richard Johansson and Pierre Nugues. 2007. Incremental dependency parsing using online learning. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL2007)*, pages 1134–1138.

Yuki Kamiya, Tomohiro Ohno, and Shigeki Matsubara. 2010. Coherent back-channel feedback tagging of in-car spoken dialogue corpus. In *Proceedings of the 11th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL2010)*, pages 205–208.

Hideki Kashioka and Takehiko Maruyama. 2004. Segmentation of semantic units in Japanese monologues. In *Proceedings of International Conference on Speech and Language Technology and Oriental COCOSDA (ICSLT/O-COCOSDA2004)*, pages 87–92.

Yoshihide Kato, Shigeki Matsubara, Katsuhiko Toyama, and Yasuyoshi Inagaki. 2005. Incremental dependency parsing based on headed context-free grammar. *Systems and Computers in Japan*, 36:63–77.

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

Shigeki Matsubara, Akira Takagi, Nobuo Kawaguchi, and Yasuyoshi Inagaki. 2002. Bilingual spoken monologue corpus for simultaneous machine interpretation research. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC2002)*, pages 153–159.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.

Tomohiro Ohno, Masaki Murata, and Shigeki Matsubara. 2009. Linefeed insertion into Japanese spoken monologue for captioning. In *Proceedings of Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL-IJCNLP2009)*, pages 531–539.

Matthias Paulik and Alex Waibel. 2010. Spoken language translation from parallel speech audio: simultaneous interpretation as SLT training data. In *Proceedings of the 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP2010)*, pages 5210–5213.

Koichiro Ryu, Shigeki Matsubara, and Yasuyoshi Inagaki. 2006. Simultaneous English-Japanese spoken language translation based on incremental dependency parsing and transfer. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-2006)*, pages 683–690.

Kiyotaka Uchimoto, Masaki Murata, Satoshi Sekine, and Hitoshi Isahara. 2000a. Dependency model using posterior context. *Journal of Natural Language Processing*, 7(5):3–18. (In Japanese).

Kiyotaka Uchimoto, Masaki Murata, Satoshi Sekine, and Hitoshi Isahara. 2000b. Dependency model using posterior context. In *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT2000)*.

Le Zhang. 2013. Maximum entropy modeling toolkit for Python and C++ (online). `http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html` (accessed 2013-08-21).