IEICE TRANS. INF. & SYST., VOL.E93–D, NO.5 MAY 2010

PAPER

Improved Sequential Dependency Analysis Integrating Labeling-Based Sentence Boundary Detection

Takanobu OBA[†a)], **Takaaki HORI**[†b)], *and* **Atsushi NAKAMURA**[†c)], *Members*

SUMMARY A dependency structure interprets modification relationships between words or phrases and is recognized as an important element in semantic information analysis. With the conventional approaches for extracting this dependency structure, it is assumed that the complete sentence is known before the analysis starts. For spontaneous speech data, however, this assumption is not necessarily correct since sentence boundaries are not marked in the data. Although sentence boundaries can be detected before dependency analysis, this cascaded implementation is not suitable for online processing since it delays the responses of the application. To solve these problems, we proposed a sequential dependency analysis (SDA) method for online spontaneous speech processing, which enabled us to analyze incomplete sentences sequentially and detect sentence boundaries simultaneously. In this paper, we propose an improved SDA integrating a labeling-based sentence boundary detection (SntBD) technique based on Conditional Random Fields (CRFs). In the new method, we use CRF for soft decision of sentence boundaries and combine it with SDA to retain its online framework. Since CRF-based SntBD yields better estimates of sentence boundaries, SDA can provide better results in which the dependency structure and sentence boundaries are consistent. Experimental results using spontaneous lecture speech from the Corpus of Spontaneous Japanese show that our improved SDA outperforms the original SDA with SntBD accuracy providing better dependency analysis results.

key words: *sequential dependency analysis, labeling, CRF, sentence boundary detection*

1. Introduction

Speech recognition research has been expanded to increase the capability of practical usage. In general, speech recognition systems are designed to transcribe input speech signals faithfully. However, most speech applications require the extraction of specific phrases and/or relationships between words/phrases from a transcribed word sequence to understand the speaker's intention. In this paper, we focus on dependency structure, which is considered a basic element for semantic analysis.

Dependency analysis has been a very active research area in recent years. Several approaches such as spanning tree models and transition-based parsing have been proposed and have gained attention as methods for providing accurate analysis results [1]–[7]. Furthermore, some research has focused on the expansion of the ability to parse a sentence with a more complex structure appropriately [8]–[10]. In contrast, we have developed a dependency analysis method that includes sentence boundary detection for spontaneously spoken language, which is suitable for online applications.

Online processing is indispensable for advanced speech applications such as simultaneous speech translation, real-time captioning with speech summarization, and human-like dialog systems. For example, in the future humanoid robots may be able to nod, smile, and interrupt a user's utterances. To enable robots to perform such actions, a language analysis should be performed online for a stream of words from a speech recognizer. This type of analysis corresponds to the fact that human parsing is essentially incremental.

We have already proposed sequential dependency analysis (SDA) for online speech applications [11]. To achieve the online processing of dependency analysis, dependency parsers must provide the three functions described below.

- Analysis of incomplete sentences
  An online dependency analyzer needs to receive and parse an input sequence at a certain time without waiting for sentence ends. If such an incomplete sentence is analyzed with a conventional offline dependency parser, the result will contain many errors since the parser is designed to analyze only complete sentences. To overcome this problem, we need to consider the dependency relationships including unseen words that will subsequently be provided by the speech recognizer.
- Early determination of dependency links
  For online applications, it is useful for a partial dependency structure to be made available by determining dependency links as soon as possible. Once an input sequence is parsed, the dependency structure should be updated without any changes in the dependency links except for those with the unseen part when the next input sequence is parsed.
- Analysis detecting sentence boundaries
  An input word sequence from a standard speech recognizer does not include any sentence boundary annotations. Therefore, we usually need to realize sentence boundary detection (SntBD) before parsing the input sequence when we apply a conventional offline dependency parser. In online processing, sentence boundaries must be estimated simultaneously with dependency parsing.

Manuscript received August 24, 2009.
Manuscript revised December 21, 2009.

[†]The authors are with NTT Communication Science Laboratories, NTT Corporation, Kyoto-fu, 619–0237 Japan.
a) E-mail: oba@cslab.kecl.ntt.co.jp
b) E-mail: hori@cslab.kecl.ntt.co.jp
c) E-mail: ats@cslab.kecl.ntt.co.jp

DOI: 10.1587/transinf.E93.D.1272

Copyright © 2010 The Institute of Electronics, Information and Communication Engineers

Although some left-to-right parsing algorithms have been proposed [6], [7], they are assumed to analyze complete sentences, i.e. they are not designed for online speech applications. SDA satisfies the above three conditions and enables us to analyze incomplete sentences sequentially and detect sentence boundaries simultaneously [11]. SDA achieves online processing with an accuracy equivalent to that of offline processing in which boundary detection and dependency analysis are cascaded.

In this paper, we propose an improved SDA in which we integrate labeling-based SntBD with Conditional Random Fields (CRFs) [12] to boost both dependency and SntBD accuracies. In the new method, we use CRF for soft decision of sentence boundaries and combine it with SDA to retain its online framework. Since the CRF-based SntBD yields more reliable sentence boundaries than those provided by the original SDA, the new method outperforms the original SDA as regards both dependency and SntBD accuracies, and even CRF itself in terms of SntBD accuracy. This is the synergic effect provided through the SDA, which finds the best structure so that sentence boundaries are consistent with the dependency structure.

Some recent research has proved that labeling-based SntBD using dependency information increases the accuracy of both dependency parsing and SntBD [13], [14]. However, these techniques are not designed for online processing, that is, the sentence boundaries are re-estimated after SntBD and dependency analysis have been applied in this order. Furthermore, to extract conclusive dependency structures, dependency parsing must be applied again. The improved SDA works incrementally along an input word sequence together with labeling-based SntBD, and it provides the analysis results as soon as possible.

In this work, we apply the improved SDA specifically to Japanese dependency analysis in which dependency structure is usually defined as dependency links between Japanese phrase-like units called *bunsetsu*s but not between words. Therefore, we must detect *bunsetsu* segments for an input word sequence, before analyzing dependency structures. *Bunsetsu* and sentence boundaries are usually detected simultaneously by using a labeling method. The improved SDA utilizes such segmentation results with the detection scores.

In the proposed method, first *bunsetsu* and sentence boundaries are estimated with a labeling method. Next, while SDA parses the *bunsetsu* sequence, in which each dependency link score is weighted by a label score that indicates whether or not the last word of the modifier *bunsetsu* is located at a sentence end. Note that these analyzing steps proceed synchronously with the input sequence in online processing.

This paper is organized as follows: in Sect. 2, we describe both SDA and labeling techniques. Our proposed method is described in Sect. 3. Section 4 provides experimental results using the corpus of spontaneous Japanese (CSJ) [15]. And Sect. 5 concludes this paper.

2. Extraction of Dependency Structures

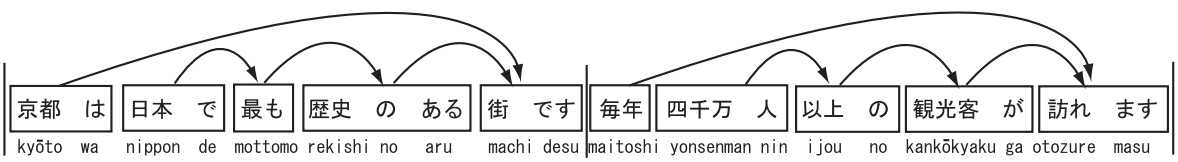
In this section, we briefly describe Japanese dependency analysis and review the original SDA we proposed for the online dependency analysis of spoken language.

2.1 Japanese Dependency Analysis

In natural language processing for a Japanese word sequence, the sequence is typically divided into *bunsetsu*s. *Bunsetsu*s are used as basic components for interpreting a given input word sequence and have the following characteristics.

- All *bunsetsu*s are continuous, that is, each word belongs to a *bunsetsu*.
- A sentence boundary necessarily corresponds to a *bunsetsu* boundary.
- A pause in speech is considered a *bunsetsu* boundary (especially in CSJ).

When a dependency analyzer receives a word sequence from a speech recognizer, the segment boundaries of the *bunsetsu*s and sentences are unknown. The segments must be decided before or while analyzing the dependency structures between *bunsetsu*s. An example of extracted dependency structures is shown in Fig. 1. The square, vertical bars and arrowed lines denote *bunsetsu*s, sentence boundaries and dependency links, respectively. When a *bunsetsu* *b* modifies another *bunsetsu* *h*, it is said that *b* links to *h* and this is represented as $b \rightarrow h$. *b* is called the modifier and *h* is called the head. The *bunsetsu* that does not modify any *bunsetsu*s in a sentence is called the sentence head. “街 です” and “訪れ ます” are the sentence heads in this figure.



In English: “Kyoto is the most historic city in Japan. More than forty million tourists visit every year.” Squares express *bunsetsu*s and vertical bars denote sentence boundaries.

Fig. 1 Dependency structure.

2.2 Sequential Dependency Analysis

Next, we describe the original SDA method after explaining a typical Japanese dependency analysis method. SDA and this typical method share a common framework. Labeling methods for obtaining segments of *bunsetsu*s and sentences are described in the next section.

2.2.1 Dependency Modeling

A dependency structure D is represented as a set of head *bunsetsu*s $h_1, h_2, \cdots, h_N$ corresponding to modifier *bunsetsu*s $B = (b_1, b_2, \cdots, b_N)$. The dependency analysis finds the most appropriate dependency structure D^* from hypothetical structures of a sentence. The most general method is based on probabilistic parsing. That is,

$$D^* = \arg\max_D P(D|B). \quad (1)$$

$P(D|B)$ is the probability for generating the structure D where *bunsetsu* sequence B is given and is calculated as

$$P(D|B) = \prod_{i=1}^{N} P(b_i \rightarrow h_{b_i} | \mathbf{\Phi}(b_i, h_{b_i}, B)) \quad (2)$$

$\mathbf{\Phi}(b_i, h_{b_i}, B)$ is a linguistic feature vector. The link score $P(b_i \rightarrow h_{b_i} | \mathbf{\Phi}(b_i, h_{b_i}, B))$ is trained using dependency parsed data.

The dependency structure usually has some constraints. One is that each *bunsetsu* other than the sentence head has only one head. Equation (2) is calculated under this constraint. Another constraint is that dependency links never cross each other. The most appropriate dependency structure given by Eq. (1) is found from such structures whose links never cross.

The Japanese dependency structure has one more constraint.

- Dependency links are only directed from left to right, and therefore the sentence head is necessarily the rightmost *bunsetsu* in a complete sentence.

In spontaneous speech, we can observe certain exceptions that do not satisfy these constraints, but they rarely occur in actual situations. We also assume that a speech recognition system with a POS tagged language model can find and remove filler *bunsetsu*s that do not have any heads.

For the experiments we describe in Sect. 4, we used the parsing algorithm proposed in [16] to find the most appropriate dependency structure. In this algorithm, dependency links are decided in order from the end to the start of the sentence. Although some efficient methods that take account of the three constraints have been proposed for Japanese dependency analysis [6], [7], [16]–[19], we have not yet found how to combine them with SDA. This will constitute future work.

Some dependency models for calculating dependency link probabilities $P(b_i \rightarrow h_{b_i} | \mathbf{\Phi}(b_i, h_{b_i}, B))$ have been proposed. Uchimoto et al. have proposed a modeling method based on Maximum Entropy [18] and also have modeled a dependency link that takes context structures into consideration [20]. Kudo et al. have proposed two types of modeling methods [19]. One is the absolute dependency model. A link probability is trained to distinguish whether a link is true or false. The other is the relative dependency model. Since this is known to be an accurate model among probabilistic dependency models for parsing Japanese dependencies, we use it in our work.

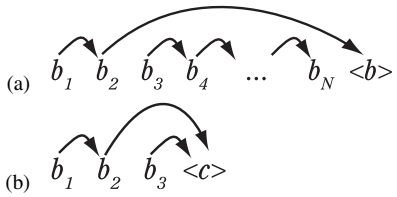


Fig. 2 Dependency structure for complete (a) and incomplete sentences (b), where (b) consists of the three *bunsetsu*s from the beginning of (a).

[Relatively dependency learning]

The dependency link score $P(b_i \rightarrow h_{b_i} | \mathbf{\Phi}(b_i, h_{b_i}, B))$ is learned to determine the correct b_i head from a number of candidates. We call this learning model a relative model.

The formulation based on the maximum entropy method is

$$P(b_i \rightarrow h_{b_i} | \mathbf{\Phi}(b_i, h_{b_i}, B)) = \frac{\exp(\boldsymbol{w} \cdot \mathbf{\Phi}(b_i, h_{b_i}, B))}{\sum_{h \in C_{b_i}} \exp(\boldsymbol{w} \cdot \mathbf{\Phi}(b_i, h, B))} \quad (3)$$

where w is a model parameter and C_{b_i} is a set of b_i head candidates, which is given based on the parsing algorithm and the dependency constraints.

2.2.2 Sequential Parsing

SDA provides online dependency analysis without changing the formulation of Eq. (1) to extract the most appropriate dependency structure. The key to this expansion is the introduction of meta symbols to detect sentence boundaries and model the dependency structure including the unseen part that is input later.

Suppose that a complete sentence B is formed by a sequence of N *bunsetsu*s, $b_1, b_2, \cdots, b_N$. The representation of the dependency structure of the complete sentence is redefined by introducing meta symbol <*b*> with the link from the sentence head as shown in Fig. 2-(a). <*b*> is utilized to detect a sentence boundary while parsing a given sequence.

On the other hand, we assume that an incomplete sentence comprises the first n *bunsetsu*s of the complete sentence, $b_1, b_2, \cdots, b_n$ $(1 \leq n \leq N-1)$, where the sequence $b_{n+1} \ldots b_N$ is unseen. Another meta symbol <*c*> is introduced to represent an arbitrary *bunsetsu* in the unseen part and an expression $b_i \rightarrow$<*c*> is defined as a dependent relationship where b_i modifies a *bunsetsu* in the unseen part.

Next we describe the SDA algorithm. The algorithm processes a given sequence *block by block*, where the *block*

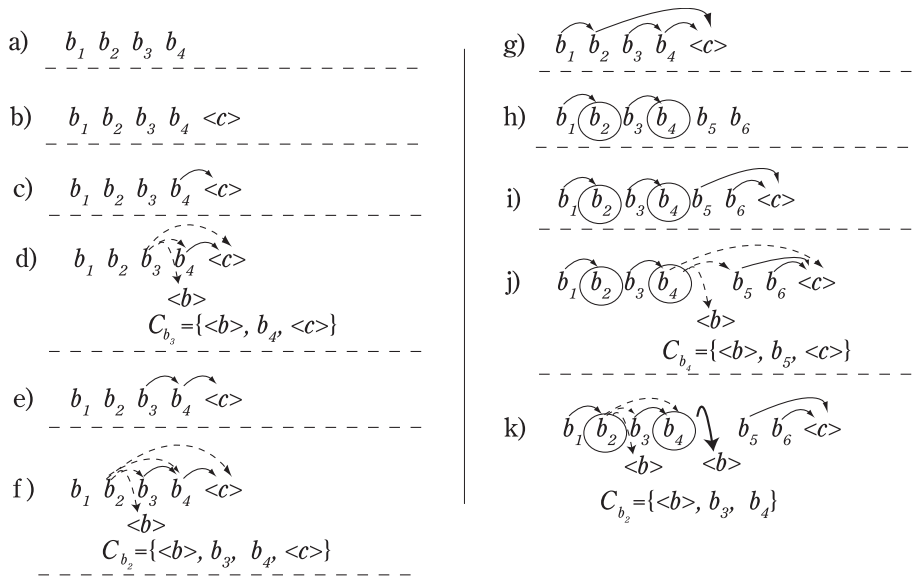


Fig. 3 Flow of sequential dependency analysis.

indicates any subsequence of the *bunsetsu* sequence to be processed. For example, most speech recognizers can output a recognition result immediately after a short or long pause in the waveform. Thus, a block can be defined as a *bunsetsu* sequence between pauses. We employed this block definition for the experiments described in this paper.

Here, we describe the flow of the analysis according to the example shown in Fig. 3. Although the parsing algorithm and the link score model that we used for our experiments are reflected in this flow, most parsing algorithms and link score models can be employed in SDA. The algorithm is summarized in Appendix.

a) Read the first block, $b_1 \ldots b_4$.
b) Add a meta symbol <*c*> to the end of the current sequence. The link for each *bunsetsu* is searched for in reverse order (b_4, b_3, b_2, b_1) according to Sekine's algorithm. <*c*> is treated in the same manner as the other regular *bunsetsu*s.
c) First the link from the last *bunsetsu* to <*c*>, i.e. $b_4 \rightarrow$ <*c*>, is extracted where we assume each *bunsetsu* has one head *bunsetsu* in subsequent *bunsetsu*s.
d) Then, b_3's head is selected from $C_{b_3} = \{<b>, b_4, <c>\}$, which is decided based on the linguistic constraints of the dependency structure. During the course of the whole analysis process, <*b*> is inserted at a point where a sentence boundary possibly exists, and it is estimated whether or not <*b*> has a link with a certain *bunsetsu*. At each position where <b> is placed, a sentence boundary is detected if at least one link with <b> is extracted. Although the position of <*b*> can be decided based on certain grammatical rules, in this work we placed <*b*> between any *bunsetsu*s. The dashed arrows represent three choices for the link with b_3's head.
e) We assume that $b_3 \rightarrow b_4$ is chosen.
f) Select b_2's head from $C_{b_2} = \{<b>, b_3, b_4, <c>\}$. The dashed arrows represent four choices for the links with b_2's head.
g) Continue the process up to b_1 as with the previous procedure.
h) Remove <*c*> and then read the next block b_5, b_6.
i) Add the meta symbol <*c*> again and then continue dependency parsing in the reverse order as $b_6, b_5, \ldots b_1$. The figure shows the result after processing b_5.
j) Continue dependency parsing for the *bunsetsu*s that were linked with <*c*> in the previous block, which correspond to b_4 and b_2. The head of b_4 is selected from $C_{b_4} = \{<b>, b_5, <c>\}$. b_6 is not in C_{b_4} because of the dependency constraint, whereby no links ever cross each other.
k) Where the head of b_4 is <*b*>, select the head of b_2 from $C_{b_2} = \{<b>, b_3, b_4\}$. C_{b_2} does not include any *bunsetsu*s further back than b_4 because there are no links across sentences.
l) Repeat the procedural steps from h) until no blocks remain.

The dependency model is trained in the same manner as the parsing algorithm, which is explained here.

The dependency model for SDA can be trained using the maximum entropy method. However, to train the model, we need a corpus that contains incomplete sentences with dependency links to <*c*>. Although such a corpus does not exist, we can generate it from a general corpus such as CSJ by substituting backward *bunsetsu*s from a certain position in the *bunsetsu* stream with <*c*>.

3. Improved SDA Integrating Labeling-Based Boundary Detection

Our proposed method integrates SDA with a labeling-based boundary detection technique, in which dependency and la-

Bs	I	Bb	I	I	O	Bb	Bb	I
えー	あのっ	今日	は	ですね	<pause>	えー	日本語	の
ē	ano	kyō	wa	desune		ē	nippon	no
Bb	I	I	Bb	I	O	Bs	I	O
係り受け解析	に	ついて	お話	します	<pause>	一般	に	<pause>
kakariukekaiseki	ni	tsuite	ohanashi	shimasu		ippan	ni	
Bb	O	Bb	I	Bb	I	Bb		
あのー	<pause>	他	の	言語	と	異なり	···	
anō		ta	no	genngo	to	kotonari		

Fig. 4 An example of labels for detecting *bunsetsu* and sentence boundaries. The first sentence means "Ah um Today's talk is about Japanese dependency analysis." The second incomplete sentence starting from '一般' means "Generally, ah, it has some differences with for other language and ···."

bel scores are used to improve both dependency and boundary accuracy in an online framework. This section briefly describes the labeling method, and then explains how the improved SDA works with the labeling method.

3.1 Boundary Detection as Labeling Problem

Bunsetsu and sentence boundary detection can be handled as a labeling problem. Each token that might be a word or a pause information symbol is given a label that denotes the role of the token. Boundary detection is the problem of detecting beginning (or ending) tokens. To detect both *bunsetsu* and sentence boundaries, we use the IOB encoding scheme [21]. We introduce the four following labels.

- Bs : This label indicated that the token is at the beginning of the sentence.
- Bb : This label indicated that the token is at the beginning of the *bunsetsu* and not at the beginning of the sentence.
- I : This label indicated that the token is a word.
- O : This label indicated that the token is an information token other than a word, such as a pause symbol.

Figure 4 shows example labels. The number of *bunsetsu*s is equal to the summation of the number of Bs and Bb. In this example, eleven *bunsetsu*s are extracted from a complete sentence and an incomplete sentence.

The label for each token can be determined automatically by a classifier based on Conditional Random Fields (CRFs).

[Conditional random field based labeling]

CRFs learn to discriminate the correct labeling sequence from the other possible sequences and thus resolve the bias problem of HMM and remove the heuristic factors in the analysis procedure included in SVM based analyzers. A CRF on $(\boldsymbol{x}, \boldsymbol{y})$ is specified by a local feature vector $\boldsymbol{f}$ and the corresponding weights $\boldsymbol{\lambda}$, where a token and a label are denoted by x and y, respectively. Each local feature vector usually consists of the state features $\boldsymbol{s}(y, \boldsymbol{x}, i)$ and the transition feature $t(y, y\prime, \boldsymbol{x}, i)$, where $y\prime$ is the label at position $i-1$. The CRF's global feature vector is given by $\boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x}) = \sum_i \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{x}, i)$. Then, the conditional probability distribution based on the CRFs is defined as

$$P_{\boldsymbol{\lambda}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp\{\boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x})\}}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x})} \tag{4}$$

where $Z_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \exp\{\boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x})\}$.

The most probable label sequence $\hat{\boldsymbol{y}}$ for an input sequence $\boldsymbol{x}$ is

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} P_{\boldsymbol{\lambda}}(\boldsymbol{y}|\boldsymbol{x}) = \boldsymbol{\lambda} \cdot \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{x}), \tag{5}$$

and can be efficiently sought using the Viterbi algorithm.

3.2 SDA with Label Score

Our proposed method works in the same manner as the original SDA, but it utilizes label scores. The sequential analyzer updates dependency links when it receives a new input block. First each token in the block is labeled Bs, Bb, I, or O according to its label score. Next the block is segmented into *bunsetsu*s at each boundary just before a token labeled Bs or Bb. However, the sentence boundaries are not yet decided in this step. They are finally decided by the improved SDA taking account of the label scores $P_{\boldsymbol{\lambda}}(y_{b_i} = \text{Bs}|B)$ and $P_{\boldsymbol{\lambda}}(y_{b_i} \neq \text{Bs}|B)$ calculated with Eq. (4), where $P_{\boldsymbol{\lambda}}(y_{b_i}) = \text{Bs}|B)$ denotes the Bs label score for the first token (word) of *bunsetsu* b_i, i.e. b_i is the beginning of the sentence, while $P_{\boldsymbol{\lambda}}(y_{b_i} \neq \text{Bs}|B)$ is the score for the first token (word) of b_i to be labeled Bb, I or O, i.e. b_i is NOT the beginning of the sentence. In dependency parsing, the proposed method employs the following dependency link score instead of the link probability $P(b_i \rightarrow h_{b_i}|\boldsymbol{\Phi}(b_i, h_{b_i}, B))$ used in the original SDA.

$$Q(b_i \rightarrow h_{b_i}|\boldsymbol{\Phi}) = \begin{cases} P_{\boldsymbol{\lambda}}(y_{b_{i+1}} = \text{Bs}|B)^{\alpha} \cdot P(b_i \rightarrow <b>|\boldsymbol{\Phi}) \\ \qquad\qquad \text{where} \quad h_{b_i} = <b> \\ P_{\boldsymbol{\lambda}}(y_{b_{i+1}} \neq \text{Bs}|B)^{\alpha} \cdot P(b_i \rightarrow h_{b_i}|\boldsymbol{\Phi}) \\ \qquad\qquad \text{where} \quad h_{b_i} \neq <b> \end{cases} \tag{6}$$

where α is the scaling weight decided experimentally by using development data. And $\boldsymbol{\Phi}(b_i, h_{b_i}, B)$ is abbreviated to $\boldsymbol{\Phi}$. In this method, if head h_{b_i} is <b>, the original dependency link score is weighted by $P_{\boldsymbol{\lambda}}(y_{b_{i+1}} = \text{Bs}|B)$. If h_{b_i} is not <b>, the link score is weighted by $P_{\boldsymbol{\lambda}}(y_{b_{i+1}} \neq \text{Bs}|B)$. *bunsetsu* boundaries are determined simply by the labeling method before dependency parsing because the labeling method is sufficiently accurate to detect *bunsetsu* boundaries.

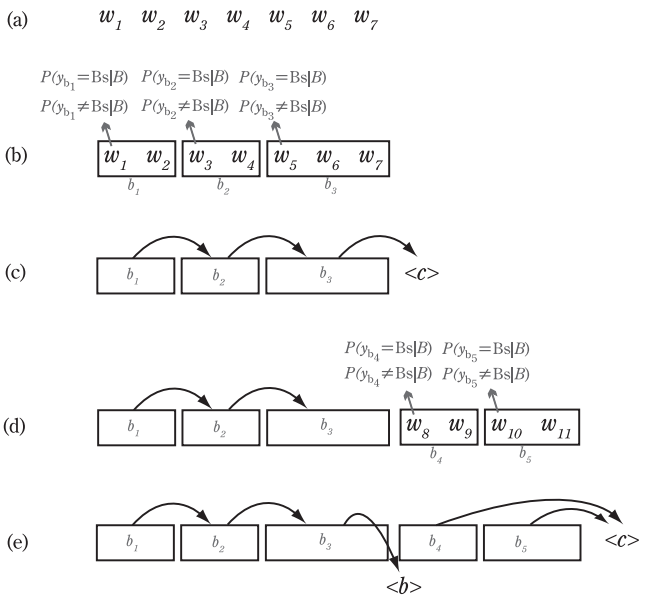


Fig. 5 Improved SDA with labeling.

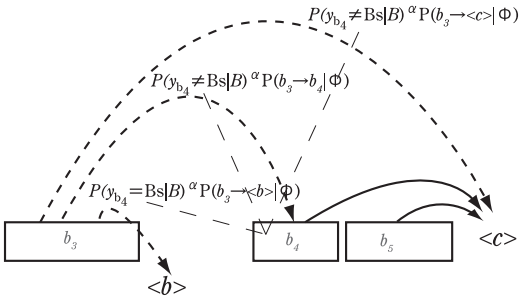


Fig. 6 Dependency link determined by combined score.

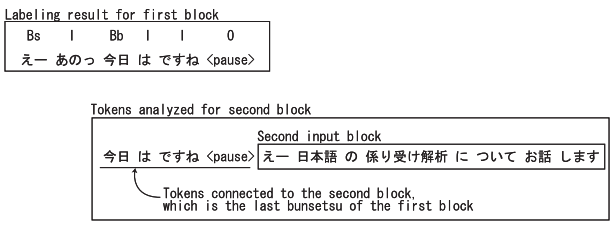


Fig. 7 Outline of incremental labeling.

Both the labeling method and SDA use similar criteria for training. However, different types of features are used. To obtain a labeling score at a position i, we generally use features obtained from some words around i. Actually, the some words have a big impact for the label score, although CRFs decide the label scores taking account of the context labels. In contrast, features obtained from long distance *bunsetsu*s are used for parameter estimation and score calculation of SDA. Therefore, by merging the labeling method and SDA, which have different characteristics, we expect them to have a complementary effect on dependency and SntBD accuracies.

In fact, our proposed method enhances SntBD accuracy. And, as a result of high-accuracy SntBD, some dependency analysis errors are recovered. If there are many head candidates, it is difficult to find the correct head. High-accuracy SntBD reduces the number of redundant head candidates because we have the constraint that dependency never links across sentences. Hence, we can expect our proposed method to eliminate some dependency analysis errors.

Figure 5 shows an example of the processing flow. Assume the first block consists of w_1, w_2, $\cdots$, w_7 as in Fig. 5 (a). If w_1, w_3, and w_5 are labeled Bs or Bb, this block can be segmented into *bunsetsu*s b_1, b_2 and b_3 as in Fig. 5 (b). Label scores $P_\lambda(y_{b_i} = \mathrm{Bs}|B)$ and $P_\lambda(y_{b_i} \neq \mathrm{Bs}|B)$ are stored for each token at the beginning of each *bunsetsu*. After adding <c> to the last *bunsetsu*, dependency parsing is performed for this *bunsetsu* sequence. For example, the dependency links are obtained as in Fig. 5 (c). After receiving the next block w_8, $\cdots$, w_{11}, these steps are performed as in Fig. 5 (d)(e). Figure 6 shows three choices for the head of b_3 where $C_{b_3} = \{<b>, b_4, <c>\}$, which are shown with dashed lines. Each link score is multiplied by the corresponding label score, and then the best head in C_{b_3} is selected.

3.3 Incremental Labeling for SDA

In sequence labeling problems, feature vector $\boldsymbol{f}$ of a position i is usually obtained from $i \pm \beta$ where β is a certain positive constant. With the improved SDA, if such a labeling process is performed for each block independently, the mislabeling risk increases, especially at the beginning of the input block, because no context information is available.

To avoid this, when labeling a new input block, we also utilize the previous input block. If the beginning of a sentence is at the beginning of the new input sequence, it would be easily detected by using context features from the previous block.

In Fig. 7, first, tokens in the first input block are labeled. This is described as the 'Labeling result for first block'. Then, we use the previous block to label the second block. We use only the last *bunsetsu* segments detected in the previous labeling result. All tokens in "Tokens analyzed for second block" are labeled.

We apply the rule that once the labels are determined they are not changed by subsequent analysis. This makes it impossible to obtain different labels in the connected part. The procedure is repeated until the end of the analysis for the last input sequence.

4. Experiment

We used the Corpus of Spontaneous Japanese (CSJ) for our experiments. The CSJ includes the speech data, the manually transcribed text, *bunsetsu* boundaries, sentence boundaries, dependency structures and pause information. We divided 189 lecture data as shown in Table 1 for model training and evaluation.

Table 1 Data sets.

	# of lectures	# of sentences	# of *bunsetsu*s	# of words
Training set	169	16,885	184,409	419,742
Evaluation set	10	1,012	11,162	26,122
Development set	10	715	8,108	19,634

Table 2 Relationships between the methods for SntBD and dependency analysis.

	detection of sentence boundaries	dependency analysis
SntBD+DA	labeling	dependency analysis of given sentences
SDA	SDA	SDA
Proposal	SDA considering label scores	

To show advantages of our proposed method, we compared the following three methods as regards dependency and SntBD accuracy.

[SntBD+DA] dependency analysis after detecting *bunsetsu* and sentence boundaries:

The labeling system divides input sequences into *bunsetsu*s and sentences simultaneously, and the dependency parser analyzes the structure between the detected *bunsetsu*s for each detected sentence. This is an offline analysis approach and the most conventional way to analyze Japanese spoken data.

[SDA] SDA after detecting only *bunsetsu* boundaries:

The labeling system is employed solely for detecting *bunsetsu* boundaries, and SDA detects sentence boundaries while simultaneously analyzing dependency structures between the detected *bunsetsu*s. This method is used online.

[Proposal] SDA considering label scores after detecting *bunsetsu* and sentence boundaries:

The scaling factor was decided so as to maximize the SntBD accuracy for the development set. The actual value of the scaling weight α was 1.2.

The relationships between these three methods are summarized as shown in Table 2.

To obtain the same *bunsetsu* extraction result, the same labeling procedure is employed with all the methods. That is, sequential labeling including the estimation of the label 'Bs' is applied incrementally as described in Sect. 3.3. In fact, almost the same result is obtained when labeling without the 'Bs' estimation and incremental process. To construct the labeling method, we used CRF++, which is a open-source tool for labeling problems [22]. The transcript texts of the evaluation set are analyzed in all our experiments.

The features employed for the CRF based labeling are surface form, part-of-speech (POS), POS-subcategory, inflection type, their combinations, which are obtained from $i \pm 3$, and bi-gram label features. We used a Gaussian prior distribution for parameter estimation and the coefficient for the parameter prior distribution was decided by using the development set.

The features employed for dependency analysis are surface form, POS, POS-subcategory, inflection type and inflection form, which are obtained from words in the header and modifier *bunsetsu*. And we also employ the flag of the beginning/ending *bunsetsu* of a sentence, the distance between the two *bunsetsu*s, pause symbol and their combination features. They conform with features used in previous reports [18], [19]. <*c*> and <*b*> were handled as surface forms in SDA.

Table 3 *Bunsetsu* boundary detection accuracy.

	Recall[%]	Precision[%]	F-value[%]
CRF	98.6	98.9	98.8

Table 4 Accuracy comparison of SntBD and dependency parsing. The word 'accuracy' is abbreviated as 'acc.'

	SntBD acc. (F-value)[%]	Dependency acc. [%]
SntBD+DA	85.5	75.6
SDA	84.7	76.0
Proposal	88.4	76.5

Table 3 shows *bunsetsu* boundary detection accuracy.

Recall denotes the ratio of correctly-detected boundaries to correct boundaries. Precision denotes the ratio of correctly-detected boundaries to all detected boundaries. F-value denotes the harmonic average of recall and precision. We find that CRFs detect *bunsetsu* boundaries very accurately.

Table 4 shows SntBD and dependency analysis accuracies. SntBD accuracies are represented by F-values. We recognized the correct dependency link when both the link extraction and the detection of the two *bunsetsu*s, which are the head and the modifier, are correctly analyzed. In addition, when calculating the dependency accuracies, we ignored the dependency links of *bunsetsu*s consisting of only filler words since they have no heads by definition.

A comparison of SntBD+DA and SDA showed that SDA outperformed SntBD+DA with dependency analysis while the SntBD accuracy of SDA was worse, since SDA prevented the boundary detection that destroyed the dependency structures.

Our proposed method outperformed the others with SntBD. Table 5 shows the recall and precision of SntBD for all the methods. CRF based labeling, that is, SntBD+DA, provided high precision, and SDA gave high recall. With our proposed method, both measures were almost the same. Labeling and SDA include complementary information and

Table 6 Examples of analysis results.

[Sample 1]

Reference	11	3		11	−1		20	−1
SntBD+DA	20				12			
SDA								
Proposal								
bunsetsu No.	1 これまでは koremadewa	2 自然な shizenna	···	10 規則に kisokuni	11 表しましたが arawashimashitaga	···	19 必要が hitsuyōga	20 あります arimasu

[Sample 2]

Reference	7		−1	10	10	−1
SntBD+DA	2					
SDA	10		10			
Proposal						
bunsetsu No.	1 先程 sakihodo	···	7 説明いたしましたけれども setsimēitashimashitakeredomo	8 それについて sorenitsuite	9 詳しく kuwashiku	10 説明します setsumeishimasu

[Sample 3]

Reference	−1	−1	−1	8
SntBD+DA		3		
SDA		4	4	
Proposal				
bunsetsu No.	1 表しています arawashiteimasu	2 第一音調指令の daiichionchōno	3 すいません suimasen	4 中国語において chūgokugonioite

Table 5 Recall and precision of SntBD.

	Recall[%]	Precision[%]
SntBD+DA	83.2	87.9
SDA	88.7	81.1
Proposal	88.1	88.6

these compensate for each other in our proposed method.

Our proposed method also slightly improved the dependency accuracy as shown in Table 4 where the difference was significant in the level of 10%. We provide some samples in which the SntBD and dependency analysis errors are corrected.

Table 6 shows head *bunsetsu* numbers. For example, in sample 1, Ref. 3 in the row of *bunsetsu* No. 2 indicates that 3rd *bunsetsu* is the head of 2nd *bunsetsu*. −1 means that the *bunsetsu* is a sentence end. Empty cells mean that estimation results are correct, in short, the head numbers are the same as the reference.

In sample 1, CRF does not correctly detect the sentence boundary between the 11th and 12th *bunsetsu*s. But, the sentence boundary is detected with our proposed method, since it is also detected correctly by SDA. Both SDA and our proposed method also correctly estimate the head of 1st *bunsetsu*. This is an example of a dependency analysis error being corrected by reducing the number of head candidates by finding the true sentence boundary.

In contrast, both SntBD and dependency analysis errors of SDA are corrected by SntBD+DA in sample 2. The above two examples indicate that SDA and the labeling method can complement each other in our proposed method.

In sample 3, a speaker starts to make a new sentence from the 2nd *bunsetsu* but soon stops. Then the speaker apologizes for this in the 3rd *bunsetsu* and restarts anther sentence from the 4th *bunsetsu*. So the 1st, 2nd and 3rd *bunsetsu*s are sentence ends. We know, since the 2nd *bunsetsu* is a fragment of a sentence that it is difficult to estimate it as a sentence end.

SntBD+DA could correctly answer that the 3rd *bunsetsu* was a sentence end, while SDA made a mis-estimation. Neither method could correctly estimate the 2nd *bunsetsu* as a sentence end. However, our proposed method could provide correct answers for both *bunsetsu*s. This is because SDA has the potential to estimate the 2nd *bunsetsu* as a sentence end correctly. But unfortunately a mistake with respect to the 3rd *bunsetsu* caused a mistake for 2nd *bunsetsu*. Our proposed method correctly estimated the 3rd *bunsetsu* by benefiting from CRF, and also correctly estimated the 2nd *bunsetsu* as a sentence end by benefiting from SDA.

5. Conclusion

For the online processing of speech, all processes need to be undertaken using incremental and sequential procedures. Specifically, we focused on dependency parsing and a previously proposed SDA technique that detected sentence boundaries simultaneously. Furthermore, in this paper, we expanded SDA to extract Japanese dependency structures more accurately by improving SntBD accuracy. To improve the SntBD accuracy, we used the sentence boundary score

provided by a labeling method while analyzing the dependency structures. Our experiments showed that our proposed method detected sentence boundaries more accurately than the single use of the SDA and CRF based labeling method. We also found that the high-accuracy SntBD of our proposed method could improve dependency accuracy by preventing the estimation of redundant dependency links.

References

[1] M. Iwatate, M. Asahara, and Y. Matsumoto, "Japanese dependency parsing using a tournament model," Proc. 22nd International Conference on Computational Linguistics (Coling 2008), pp.361–368, Manchester, UK, Coling 2008 Organizing Committee, Aug. 2008.
[2] R. McDonald, K. Crammer, and F. Pereira, "Online large-margin training of dependency parsers," Proc. ACL, pp.91–98, 2005.
[3] J. Hall, J. Nivre, and J. Nilsson, "Discriminative classifiers for deterministic dependency parsing," Proc. COLING/ACL, pp.316–323, 2006.
[4] M. Schiehlen and K. Spranger, "Global learning of labeled dependency trees," Proc. EMNLP-CoNLL, pp.1156–1160, 2007.
[5] H. Yamada and Y. Matsumoto, "Statistical dependency analysis with support vector machines," Proc. IWPT, pp.195–206, 2003.
[6] S. Mori, M. Nishimura, N. Itoh, S. Ogino, and H. Watanabe, "A stochastic parser based on a structural word prediction model," Proc. 18th Conference on Computational Linguistics (Coling 2000), pp.558–564, 2000.
[7] M. Sassano, "Linear-time dependency analysis for Japanese," Proc. 20th International Conference on Computational Linguistics (Coling 2004), pp.8–14, Association for Computational Linguistics, Morristown, NJ, USA, 2004.
[8] R. McDonald and F. Pereira, "Online learning of approximate dependency parsing algorithms," Proc. EACL, pp.81–88, 2006.
[9] J. Nivre and J. Nilsson, "Pseudo-projective dependency parsing," Proc. ACL, pp.99–106, 2005.
[10] J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov, "Labeled pseudo-projective dependency parsing with support vector machines," Proc. CoNLL, pp.221–225, 2006.
[11] T. Oba, T. Hori, and A. Nakamura, "Sequential dependency analysis for online spontaneous speech processing," Speech Commun., vol.50, pp.616–625, July 2008.
[12] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," Proc. International Conference on Machine Learning, pp.282–289, 2001.
[13] B. Roark, Y. Liu, M. Harper, R. Stewart, M. Lease, M. Snover, I. Shafran, B. Dorr, J. Hale, A. Krasnyanskaya, and L. Yung, "Reranking for sentence boundary detection in conversational speech," Proc. ICASSP, 2006.
[14] K. Shitaoka, K. Uchimoto, T. Kawahara, and H. Isahara, "Dependency structure analysis and sentence boundary detection in spontaneous Japanese," Proc. International Conference on Computational Linguistics, pp.1107–1113, 2004.
[15] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of Japanese," Proc. LREC2000, pp.947–952, 2000.
[16] S. Sekine, K. Uchimoto, and H. Isahara, "Backward beam search algorithm for dependency analysis of Japanese," Proc. ACL, pp.754–760, 2000.
[17] T. Kudo and Y. Matsumoto, "Japanese dependency analysis using cascaded chunking," Proc. CoNLL, pp.63–69, 2002.
[18] K. Uchimoto, S. Sekine, and H. Isahara, "Japanese dependency structure analysis based on maximum entropy models," Proc. EACL, pp.196–203, 1999.
[19] T. Kudo and Y. Matsumoto, "Japanese dependency parsing using relative preference of dependency," IPSJ SIG Technical Report, 2004-NL-162, pp.205–212, 2004.
[20] "Dependency model using posterior context,"
[21] L. Ramshaw and M. Marcus, "Text chunking using transformation-based learning," Proc. Third Workshop on Very Large Corpora, ed., D. Yarovsky and K. Church, pp.82–94, Association for Computational Linguistics, Somerset, New Jersey, 1995.
[22] T. Kudo, "CRF++." http://crfpp.sourceforge.net/

Appendix: Sequential Dependency Analysis Algorithm

We describe the pseudo code of the sequential dependency analysis method below. As shown in the code, the algorithm yields a dependency analysis result $\hat{h}$ after processing each block. Lines 11 to 28 correspond to Sekine's parsing algorithm. Each *bunsetsu* whose head is undetermined in the previous blocks, i.e., is linked with <c>, is checked again in the current block. When undertaking this check, the lines 13–27 are repeated not until $i = L + 1$, but until $i = 1$. It is also important to remove links with <c> from the current result in order to continue the analysis for the subsequent blocks.

```
 1 program sequential dependency analysis
 2 variables
 3   h, ĥ, g : candidate of dependency structure
 4   H, G : set of candidates
 5   L, m, i : integer
 6 begin
 7   L = 0
 8   ĥ = φ
 9   while the next block exists do
10     read a block consisting of m bunsetsus, \
         and let them be b_{L+1}, b_{L+2}, ..., b_{L+m}
11     add b_{L+m} → <c> to ĥ
12     H = {ĥ}
13     for i = L + m − 1 to 1 do
14       G = φ
15       for each h in H
16         if b_i's head is undetermined then
17           prepare C_{b_i} ⊆ {<b>, b_{i+1}, ..., b_{L+m}, <c>}
18           for each v in C_{b_i} do
19             generate g by copying h
20             add b_i → v to g
21             insert g into G
22           end for
23         end if
24       end for
25       prune candidates in G by keeping \
                        the top K candidates
26       H = G
27     end for
28     ĥ = argmax_{h∈H} P(b_i → h|b_1, ..., b_{L+m})
29     remove links with <c> from ĥ
30     L = L + m
31   end while
32   add v → <b> to ĥ for any v which head is undermined
33 end
```

Takanobu Oba received B.E. and M.E. degrees from Tohoku University, Sendai, Japan, in 2002 and 2004, respectively. Since 2004, he has been engaged in research on spoken language processing at the NTT Communication Science Laboratories, Kyoto, Japan. He received the 25th Awaya Kiyoshi Science Promotion Award from the Acoustical Society of Japan (ASJ) in 2008. He is a member of the Institute of Electrical and Electronics Engineers (IEEE) and the ASJ.

Takaaki Hori received B.E. and M.E. degrees in electrical and information engineering from Yamagata University, Yonezawa, Japan, in 1994 and 1996, respectively, and was awarded a Ph.D. in system and information engineering by Yamagata University in 1999. Since 1999, he has been engaged in research on spoken language processing at the NTT Cyber Space Laboratories, Yokosuka, Japan. He was a Visiting Scientist at the Massachusetts Institute of Technology, Cambridge, from 2006 to 2007. He received the 22nd Awaya Kiyoshi Science Promotion Award from the Acoustical Society of Japan (ASJ) in 2005. He is a member of the Institute of Electrical and Electronics Engineers (IEEE) and the ASJ.

Atsushi Nakamura received the B.E., M.E., and Dr.Eng. degrees from Kyushu University, Fukuoka, Japan, in 1985, 1987 and 2001, respectively. In 1987, he joined Nippon Telegraph and Telephone Corporation (NTT), where he engaged in the research and development of network service platforms, including studies on application of speech processing technologies into network services, at Musashino Electrical Communication Laboratories, Tokyo, Japan. From 1994 to 2000, he was with Advanced Telecommunications Research (ATR) Institute, Kyoto, Japan, as a Senior Researcher, working on the research of spontaneous speech recognition, construction of spoken language database and development of speech translation systems. Since April, 2000, he has been with NTT Communication Science Laboratories, Kyoto, Japan. His research interests include acoustic modeling of speech, speech recognition and synthesis, spoken language processing systems, speech production and perception, computational phonetics and phonology, and application of learning theories to signal analysis and modeling. Dr. Nakamura is a senior member of the IEEE and a member of the Acoustical Society of Japan (ASJ). Also he serve as a Vice Chair of the IEEE Signal Processing Society Kansai Chapter. He received the IEICE Paper Award, and the Telecom-technology Award of The Telecommunications Advancement Foundation, in 2004 and 2006, respectively.