Neural Modeling of Multi-Predicate Interactions for Japanese Predicate Argument Structure Analysis

Hiroki Ouchi[1,2] **Hiroyuki Shindo**[1,2] **Yuji Matsumoto**[1,2]
[1] Nara Institute of Science and Technology
[2] RIKEN Center for Advanced Intelligence Project (AIP)
{ ouchi.hiroki.nt6, shindo, matsu }@is.naist.jp

Abstract

The performance of Japanese predicate argument structure (PAS) analysis has improved in recent years thanks to the joint modeling of interactions between multiple predicates. However, this approach relies heavily on syntactic information predicted by parsers, and suffers from error propagation. To remedy this problem, we introduce a model that uses *grid-type recurrent neural networks*. The proposed model automatically induces features sensitive to *multi-predicate interactions* from the word sequence information of a sentence. Experiments on the NAIST Text Corpus demonstrate that without syntactic information, our model outperforms previous syntax-dependent models.

1 Introduction

Predicate argument structure (PAS) analysis is a basic semantic analysis task, in which systems are required to identify the semantic units of a sentence, such as *who did what to whom*. In pro-drop languages such as Japanese, Chinese and Italian, arguments are often omitted in text, and such *argument omission* is regarded as one of the most problematic issues facing PAS analysis (Iida and Poesio, 2011; Sasano and Kurohashi, 2011; Hangyo et al., 2013).

In response to the argument omission problem, in Japanese PAS analysis, a joint model of the interactions between multiple predicates has been gaining popularity and achieved the state-of-the-art results (Ouchi et al., 2015; Shibata et al., 2016). This approach is based on the linguistic intuition that the predicates in a sentence are semantically related to each other, and capturing this relation can be useful for PAS analysis. In the example sentence in Figure 1, the word "男 $_i$ (man$_i$)" is the accusative argument of the predicate "逮捕した (arrested)" and is shared by the other predicate "逃走した (escaped)" as its nominative argument. Considering the semantic relation between "逮捕した (arrested)" and "逃走した (escaped)", we intuitively know that the person arrested by someone is likely to be the escaper. That is, information about one predicate-argument relation could help to identify another predicate-argument relation.

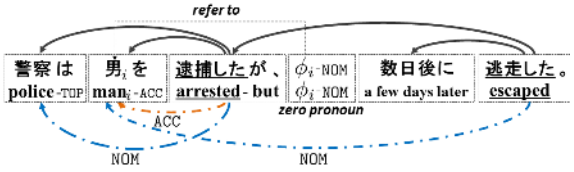

Figure 1: Example of Japanese PAS. The upper edges denote dependency relations, and the lower edges denote case arguments. "NOM" and "ACC" denote the nominative and accusative arguments, respectively. "ϕ_i" is a *zero pronoun*, referring to the *antecedent* "男 $_i$ (man$_i$)".

However, to model such *multi-predicate interactions*, the joint approach in the previous studies relies heavily on syntactic information, such as part-of-speech (POS) tags and dependency relations predicted by POS taggers and syntactic parsers. Consequently, it suffers from error propagation caused by pipeline processing.

To remedy this problem, we propose a neural model which automatically induces features sensitive to multi-predicate interactions exclusively from the word sequence information of a sentence. The proposed model takes as input all predicates and their argument candidates in a sentence at a time, and captures the interactions using grid-type recurrent neural networks (Grid-RNN) without syntactic information.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1591–1600
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1146

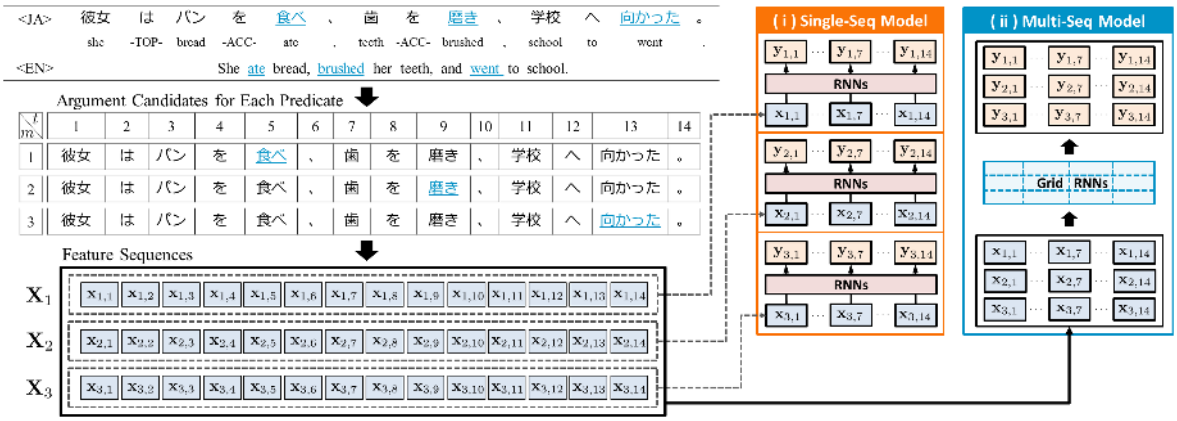

Figure 2: Overview of neural models: (i) *single-sequence* and (ii) *multi-sequence* models.

In this paper, we first introduce a basic model that uses RNNs. This model independently estimates the arguments of each predicate without considering multi-predicate interactions (Sec. 3). Then, extending this model, we propose a neural model that uses Grid-RNNs (Sec. 4).

Performing experiments on the NAIST Text Corpus (Iida et al., 2007), we demonstrate that even without syntactic information, our neural models outperform previous syntax-dependent models (Imamura et al., 2009; Ouchi et al., 2015). In particular, the neural model using Grid-RNNs achieved the best result. This suggests that the proposed grid-type neural architecture effectively captures multi-predicate interactions and contributes to performance improvements. [1]

2 Japanese Predicate Argument Structure Analysis

2.1 Task Description

In Japanese PAS analysis, arguments are identified that each fulfills one of the three major case roles, *nominative* (NOM), *accusative* (ACC) and *dative* (DAT) cases, for each predicate. Arguments can be divided into the following three categories according to the positions relative to their predicates (Hayashibe et al., 2011; Ouchi et al., 2015):

Dep: Arguments that have direct syntactic dependency on the predicate.

Zero: Arguments referred to by zero pronouns within the same sentence that have no direct syntactic dependency on the predicate.

Inter-Zero: Arguments referred to by zero pronouns outside of the same sentence.

For example, in Figure 1, the nominative argument "警察 (police)" for the predicate "逮捕した (arrested)" is regarded as a *Dep* argument, because the argument has a direct syntactic dependency on the predicate. By contrast, the nominative argument "男 $_i$ (man$_i$)" for the predicate "逃走した (escaped)" is regarded as a *Zero* argument, because the argument has no direct syntactic dependency on the predicate.

In this paper, we focus on the analysis for these intra-sentential arguments, i.e., *Dep* and *Zero*. In order to identify inter-sentential arguments (*Inter-Zero*), a much broader space must be searched (e.g., the whole document), resulting in a much more complicated analysis than intra-sentential arguments.[2] Owing to this complication, Ouchi et al. (2015) and Shibata et al. (2016) focused exclusively on intra-sentential argument analysis. Following this trend, we also restrict our focus to intra-sentential argument analysis.

2.2 Challenging Problem

Arguments are often omitted in Japanese sentences. In Figure 1, ϕ_i represents the omitted argument, called the *zero pronoun*. This zero pronoun ϕ_i refers to "男 $_i$ (man$_i$)". In Japanese PAS analysis, when an argument of the target predicate is omitted, we have to identify the antecedent of the omitted argument (i.e., the *Zero* argument).

The analysis of such *Zero* arguments is much more difficult than that for *Dep* arguments, owing to the lack of direct syntactic dependencies. For *Dep* arguments, the syntactic dependency between an argument and its predicate is a strong clue. In the sentence in Figure 1, for the predi-

[1] Our source code is publicly available at https://github.com/hiroki13/neural-pasa-system

[2] The F-measure remains 10-20% (Taira et al., 2008; Imamura et al., 2009; Sasano and Kurohashi, 2011).

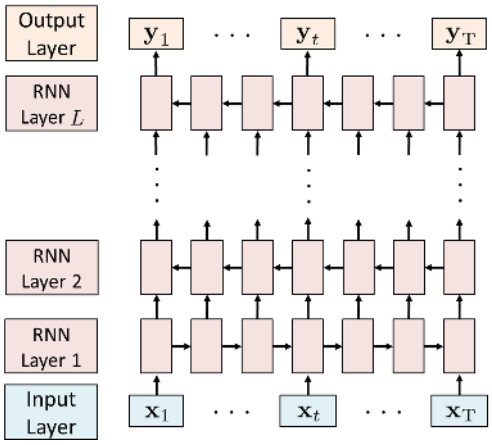

Figure 3: Overall architecture of the single-sequence model. This model consists of three components: (i) Input Layer, (ii) RNN Layer and (iii) Output Layer.

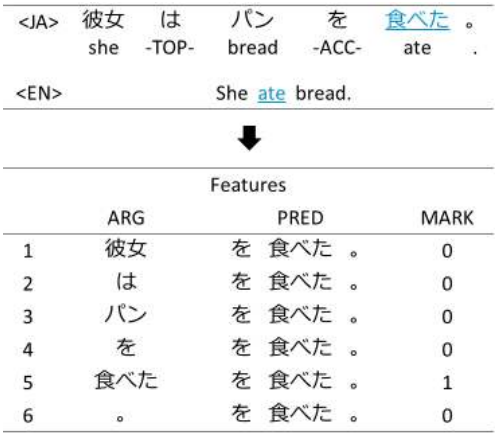

Figure 4: Example of feature extraction. The underlined word is the target predicate. From the sentence "彼女はパンを食べた。(She ate bread.)", three types of features are extracted for the target predicate "食べた (ate)".

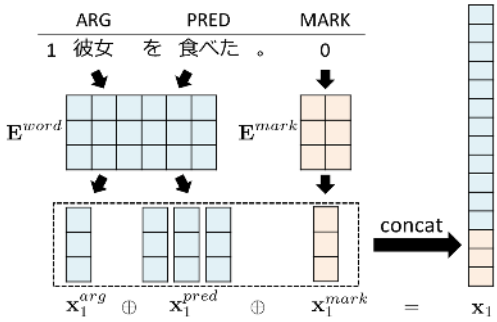

Figure 5: Example of the process of creating a feature vector. The extracted features are mapped to each vector, and all the vectors are concatenated into one feature vector.

cate "逮捕した (arrested)", the nominative argument is "警察 (police)". This argument is easily identified by relying on the syntactic dependency. By contrast, because the nominative argument "男$_i$ (man$_i$)" has no syntactic dependency on its predicate "逃走した (escaped)", we must rely on other information to identify the zero argument.

As a solution to this problem, we exploit two kinds of information: (i) the context of the entire sentence, and (ii) multi-predicate interactions. For the former, we introduce *single-sequence model* that induces context-sensitive representations from a sequence of argument candidates of a predicate. For the latter, we introduce *multi-sequence model* that induces predicate-sensitive representations from multiple sequences of argument candidates of all predicates in a sentence (shown in Figure 2).

3 Single-Sequence Model

The single-sequence model exploits stacked bidirectional RNNs (Bi-RNN) (Schuster and Paliwal, 1997; Graves et al., 2005, 2013; Zhou and Xu, 2015). Figure 3 shows the overall architecture, which consists of the following three components:

Input Layer: Map each word to a feature vector representation.

RNN Layer: Produce high-level feature vectors using Bi-RNNs.

Output Layer: Compute the probability of each case label for each word using the softmax function.

In the following subsections, we describe each of these three components in detail.

3.1 Input Layer

Given an input sentence $w_{1:T} = (w_1, \cdots, w_T)$ and a predicate p, each word w_t is mapped to a feature representation $\mathbf{x}_t$, which is the concatenation ($\oplus$) of three types of vectors:

$$\mathbf{x}_t = \mathbf{x}_t^{arg} \oplus \mathbf{x}_t^{pred} \oplus \mathbf{x}_t^{mark} \quad (1)$$

where each vector is based on the following atomic features inspired by Zhou and Xu (2015):

`ARG`**:** Word index of each word.

`PRED`**:** Word index of the target predicate and the words around the predicate.

`MARK`**:** Binary index that represents whether or not the word is the predicate.

Figure 4 presents an example of the atomic features. For the ARG feature, we extract a word index $x^{word} \in \mathcal{V}$ for each word. Similarly, for the PRED feature, we extract each word index x^{word} for the C words taking the target predicate at the center, where C denotes the window size. The MARK feature $x^{mark} \in \{0, 1\}$ is a binary value that represents whether or not the word is the predicate.

Then, using feature indices, we extract feature vector representations from each embedding matrix. Figure 5 shows the process of creating the feature vector $\mathbf{x}_1$ for the word w_1 "彼女 (she)". We set two embedding matrices: (i) a word embedding matrix $\mathbf{E}^{word} \in \mathbb{R}^{d_{word} \times |\mathcal{V}|}$, and (ii) a mark embedding matrix $\mathbf{E}^{mark} \in \mathbb{R}^{d_{mark} \times 2}$. From each embedding matrix, we extract the corresponding column vectors and concatenate them as a feature vector $\mathbf{x}_t$ based on Eq. 1.

Each feature vector $\mathbf{x}_t$ is multiplied with a parameter matrix $\mathbf{W}_x$:

$$\mathbf{h}_t^{(0)} = \mathbf{W}_x \, \mathbf{x}_t \tag{2}$$

The vector $\mathbf{h}_t^{(0)}$ is given to the first RNN layer as input.

3.2 RNN Layer

In the RNN layers, feature vectors are updated recurrently using Bi-RNNs. Bi-RNNs process an input sequence in a left-to-right manner for odd-numbered layers and in a right-to-left manner for even-numbered layers. By stacking these layers, we can construct the deeper network structures.

Stacked Bi-RNNs consist of L layers, and the hidden state in the layer $\ell \in (1, \cdots, L)$ is calculated as follows:

$$\mathbf{h}_t^{(\ell)} = \begin{cases} g^{(\ell)}(\mathbf{h}_t^{(\ell-1)}, \, \mathbf{h}_{t-1}^{(\ell)}) & (\ell = \text{odd}) \\ g^{(\ell)}(\mathbf{h}_t^{(\ell-1)}, \, \mathbf{h}_{t+1}^{(\ell)}) & (\ell = \text{even}) \end{cases} \tag{3}$$

Both of the odd- and even-numbered layers receive $\mathbf{h}_t^{(\ell-1)}$, the t-th hidden state of the $\ell-1$ layer, as the first input of the function $g^{(\ell)}$, which is an arbitrary function [3]. For the second input of $g^{(\ell)}$, odd-numbered layers receive $\mathbf{h}_{t-1}^{(\ell)}$, whereas even-numbered layers receive $\mathbf{h}_{t+1}^{(\ell)}$. By calculating the hidden states until the L-th layer, we obtain a hidden state sequence $\mathbf{h}_{1:\mathrm{T}}^{(L)} = (\mathbf{h}_1^{(L)}, \cdots, \mathbf{h}_{\mathrm{T}}^{(L)})$. Using each vector $\mathbf{h}_t^{(L)}$, we calculate the probability of case labels for each word in the output layer.

[3] In this work, we used the Gated Recurrent Unit (GRU) (Cho et al., 2014) as the function $g^{(\ell)}$.

3.3 Output Layer

For the output layer, multi-class classification is performed using the softmax function:

$$\mathbf{y}_t = \text{softmax}(\mathbf{W}_y \, \mathbf{h}_t^{(L)})$$

where $\mathbf{h}_t^{(L)}$ denotes a vector representation propagated from the last RNN layer (Fig 3). Each element of $\mathbf{y}_t$ is a probability value corresponding to each label. The label with the maximum probability among them is output as a result. In this work, we set five labels: NOM, ACC, DAT, PRED, null. PRED is the label for the predicate, and null denotes a word that does not fulfill any case role.

4 Multi-Sequence Model

Whereas the single-sequence model assumes independence between predicates, the multi-sequence model assumes *multi-predicate interactions*. To capture such interactions between all predicates in a sentence, we extend the single-sequence model to the multi-sequence model using Grid-RNNs (Graves and Schmidhuber, 2009; Kalchbrenner et al., 2016). Figure 6 presents the overall architecture for the multi-sequence model, which consists of three components:

Input Layer: Map words to M sequences of feature vectors for M predicates.

Grid Layer: Update the hidden states over different sequences using Grid-RNNs.

Output Layer: Compute the probability of each case label for each word using the softmax function.

In the following subsections, we describe these three components in detail.

4.1 Input Layer

The multi-sequence model takes as input a sentence $w_{1:\mathrm{T}} = (w_1, \cdots, w_{\mathrm{T}})$ and all predicates $\{p_m\}_1^M$ in the sentence. For each predicate p_m, the input layer creates a sequence of feature vectors $\mathbf{X}_m = (\mathbf{x}_{m,1}, \cdots, \mathbf{x}_{m,\mathrm{T}})$ by mapping each input word w_t to a feature vector $\mathbf{x}_{m,t}$ based on Eq 1. That is, for M predicates, M sequences of feature vectors $\{\mathbf{X}_m\}_1^M$ are created.

Then, using Eq. 2, each feature vector $\mathbf{x}_{m,t}$ is mapped to $\mathbf{h}_{m,t}^{(0)}$, and a feature sequence is created for a predicate p_m, i.e., $\mathbf{H}_m^{(0)} = (\mathbf{h}_{m,1}^{(0)}, \cdots, \mathbf{h}_{m,\mathrm{T}}^{(0)})$. Consequently, for M predicates, we obtain M feature sequences $\{\mathbf{H}_m^{(0)}\}_1^M$.

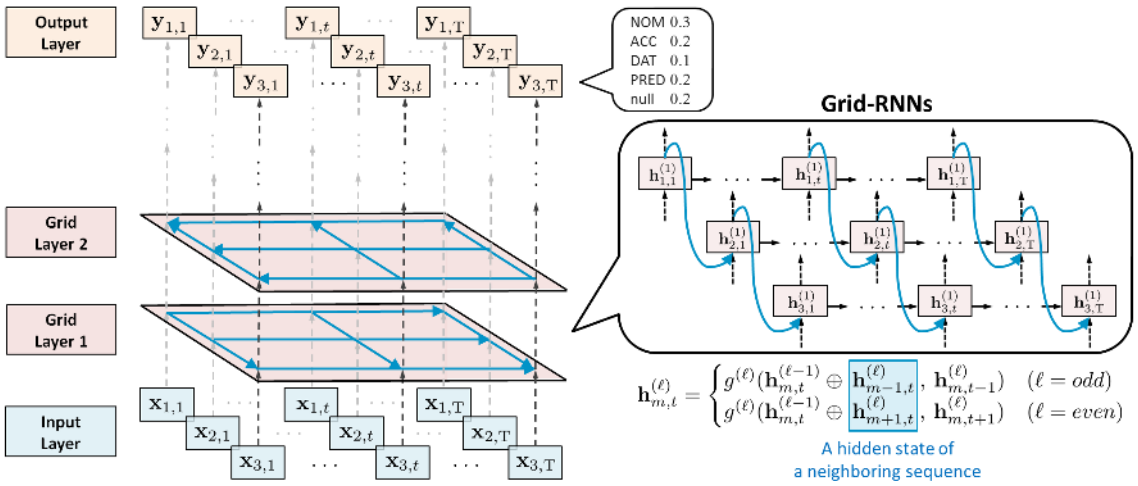

Figure 6: Overall architecture of the multi-sequence model: an example of three sequences.

4.2 Grid Layer

Inter-Sequence Connections

For the grid layers, we use Grid-RNNs to propagate the feature information over the different sequences (*inter-sequence connections*). The figure on the right in Figure 6 shows the first grid layer. The hidden state is recurrently calculated from the upper-left ($m = 1, t = 1$) to the lower-right ($m = M, t = \mathrm{T}$).

Formally, in the ℓ-th layer, the hidden state $\mathbf{h}^{(\ell)}_{m,t}$ is calculated as follows:

$$\mathbf{h}^{(\ell)}_{m,t} = \begin{cases} g^{(\ell)}(\mathbf{h}^{(\ell-1)}_{m,t} \oplus \mathbf{h}^{(\ell)}_{m-1,t}, \mathbf{h}^{(\ell)}_{m,t-1}) & (\ell = \text{odd}) \\ g^{(\ell)}(\mathbf{h}^{(\ell-1)}_{m,t} \oplus \mathbf{h}^{(\ell)}_{m+1,t}, \mathbf{h}^{(\ell)}_{m,t+1}) & (\ell = \text{even}) \end{cases}$$

This equation is similar to Eq. 3. The main difference is that the hidden state of a neighboring sequence, $\mathbf{h}^{(\ell)}_{m-1,t}$ (or $\mathbf{h}^{(\ell)}_{m+1,t}$), is concatenated ($\oplus$) with the hidden state of the previous ($\ell - 1$) layer, $\mathbf{h}^{(\ell-1)}_{m,t}$, and is taken as input of the function $g^{(\ell)}$.

In the figure on the right in Figure 6, the blue curved lines represent the inter-sequence connections. Taking as input the hidden states of neighboring sequences, the network propagates feature information over multiple sequences (i.e., predicates). By calculating the hidden states until the L-th layer, we obtain M sequences of the hidden states, i.e., $\{\mathbf{H}^{(L)}_m\}^M_1$, in which $\mathbf{H}^{(L)}_m = (\mathbf{h}^{(L)}_{m,1}, \cdots, \mathbf{h}^{(L)}_{m,\mathrm{T}})$.

Residual Connections

As more layers are stacked, it becomes more difficult to learn the model parameters, owing to various challenges such as the vanishing gradient problem (Pascanu et al., 2013). In this work, we integrate residual connections (He et al., 2015; Wu et al., 2016) with our networks to form connections between layers. Specifically, the input vector $\mathbf{h}^{(\ell-1)}_{m,t}$ of the ℓ-th layer is added to the output vector $\mathbf{h}^{(\ell)}_{m,t}$. Residual connections can also be applied to the single-sequence model. Thus, we can perform experiments on both models with/without residual connections.

4.3 Output Layer

As with the single-sequence model, we use the softmax function to calculate the probability of the case labels of each word w_t for each predicate p_m:

$$\mathbf{y}_{m,t} = \text{softmax}(\mathbf{W}_y\, \mathbf{h}^{(L)}_{m,t})$$

where $\mathbf{h}^{(L)}_{m,t}$ is a hidden state vector calculated in the last grid layer.

5 Related Work

5.1 Japanese PAS Analysis Approaches

Existing approaches to Japanese PAS analysis are divided into two categories: (i) the *pointwise approach* and (ii) the *joint approach*. The pointwise approach involves estimating the score of each argument candidate for one predicate, and then selecting the argument candidate with the maximum score as an argument (Taira et al., 2008; Imamura et al., 2009; Hayashibe et al., 2011; Iida et al., 2016). The joint approach involves scoring all the predicate-argument combinations in one sentence, and then selecting the combination with the highest score (Yoshikawa et al., 2011; Sasano and Kurohashi,

2011; Ouchi et al., 2015; Shibata et al., 2016). Compared with the pointwise approach, the joint approach achieves better results.

5.2 Multi-Predicate Interactions

Ouchi et al. (2015) reported that it is beneficial to Japanese PAS analysis to capture the interactions between all predicates in a sentence. This is based on the linguistic intuition that the predicates in a sentence are semantically related to each other, and that the information regarding this semantic relation can be useful for PAS analysis.

Similarly, in semantic role labeling (SRL), Yang and Zong (2014) reported that their reranking model, which captures the multi-predicate interactions, is effective for the English constituent-based SRL task (Carreras and Màrquez, 2005). Taking this a step further, we propose a neural architecture that effectively models the multi-predicate interactions.

5.3 Neural Approaches

Japanese PAS

In recent years, several attempts have been made to apply neural networks to Japanese PAS analysis (Shibata et al., 2016; Iida et al., 2016)[4]. In Shibata et al. (2016), a feed-forward neural network is used for the score calculation part of the joint model proposed by Ouchi et al. (2015). In Iida et al. (2016), multi-column convolutional neural networks are used for the zero anaphora resolution task.

Both models exploit syntactic and selectional preference information as the atomic features of neural networks. Overall, the use of neural networks has resulted in advantageous performance levels, mitigating the cost of manually designing combination features. In this work, we demonstrate that even without such syntactic information, our neural models can realize comparable performance exclusively using the word sequence information of a sentence.

English SRL

Some neural models have achieved high performance without syntactic information in English SRL. Collobert et al. (2011) and Zhou and Xu (2015) worked on the English constituent-based SRL task (Carreras and Màrquez, 2005) using neural networks. In Collobert et al. (2011), their model exploited a convolutional neural network and achieved a 74.15% F-measure without syntactic information. In Zhou and Xu (2015), their model exploited bidirectional RNNs with linear-chain conditional random fields (CRFs) and achieved the state-of-the-art result, an 81.07% F-measure. Our models should be regarded as an extension of their model.

The main differences between Zhou and Xu (2015) and our work are: (i) constituent-based vs dependency-based argument identification and (ii) the multi-predicate consideration. For the constituent-based SRL, Zhou and Xu (2015) used CRFs to capture the IOB label dependencies, because systems are required to identify the *spans* of arguments for each predicate. By contrast, for Japanese dependency-based PAS analysis, we replaced the CRFs with the softmax function, because in Japanese, arguments are rarely adjacent to each other.[5] Furthermore, whereas the model described in Zhou and Xu (2015) predicts arguments for each predicate independently, our multi-sequence model jointly predicts arguments for all predicates in a sentence concurrently by considering the multi-predicate interactions.

6 Experiments

6.1 Experimental Settings

Dataset

We used the NAIST Text Corpus 1.5, which consists of 40,000 sentences from Japanese newspapers (Iida et al., 2007). For the experiments, we adopted standard data splits (Taira et al., 2008; Imamura et al., 2009; Ouchi et al., 2015):

Train:	Articles: Jan 1-11, Editorials: Jan-Aug
Dev:	Articles: Jan 12-13, Editorials: Sept
Test:	Articles: Jan 14-17, Editorials: Oct-Dec

We used the word boundaries annotated in the NAIST Text Corpus and the target predicates that have at least one argument in the same sentence. We did not use any external resources.

Learning

We trained the model parameters by minimizing

[4] These previous studies used unpublished datasets and evaluated the performance with different experimental settings. Consequently, we cannot compare their models with ours.

[5] In our preliminary experiment, we could not confirm the performance improvement by CRFs.

the cross-entropy loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{n}\sum_{t} \log P(y_t|x_t) + \frac{\lambda}{2}||\boldsymbol{\theta}||^2 \quad (4)$$

where θ is a set of model parameters, and the hyper-parameter λ is the coefficient governing the L2 weight decay.

Implementation Details

We implemented our neural models using a deep learning library, Theano (Bastien et al., 2012). The number of epochs was set at 50, and we reported the result of the test set in the epoch with the best F-measure from the development set. The parameters were optimized using the stochastic gradient descent method (SGD) via a mini-batch, whose size was selected from $\{2, 4, 8\}$. The learning rate was automatically adjusted using Adam (Kingma and Ba, 2014). For the L2 weight decay, the hyper-parameter λ in Eq. 4 was selected from $\{0.001, 0.0005, 0.0001\}$.

In the neural models, the number of the RNN and Grid layers were selected from $\{2, 4, 6, 8\}$. The window size C for the PRED feature (Sec. 3.1) was set at 5. Words with a frequency of 2 or more in the training set were mapped to each word index, and the remaining words were mapped to the unknown word index. The dimensions d_{word} and d_{mark} of the embeddings were set at 32. In the single-sequence model, the parameters of GRUs were set at 32×32. In the multi-sequence model, the parameters of GRUs related to the input values were set at 64×32, and the remaining were set at 32×32. The initial values of all parameters were sampled according to a uniform distribution from $[-\frac{\sqrt{6}}{\sqrt{row+col}}, \frac{\sqrt{6}}{\sqrt{row+col}}]$, where row and col are the number of rows and columns of each matrix, respectively.

Baseline Models

We compared our models to existing models in previous works (Sec. 5.1) that use the NAIST Text Corpus 1.5. As a baseline for the pointwise approach, we used the pointwise model[6] proposed in Imamura et al. (2009). In addition, as a baseline for the joint approach, we used the model proposed in Ouchi et al. (2015). These models exploit gold annotations in the NAIST Text Corpus as POS tags and dependency relations.

[6]We compared the results of the model reimplemented by Ouchi et al. (2015).

	Dep	*Zero*	*All*
Imamura+ 09	85.06	41.65	78.15
Ouchi+ 15	86.07	44.09	79.23
Single-Seq	88.10	46.10	81.15
Multi-Seq	**88.17** †	**47.12** †	**81.42** †

Table 1: F-measures in the test set. Single-Seq is the single-sequence model, and Multi-Seq is the multi-sequence model. Imamura+ 09 is the model in Imamura et al. (2009) reimplemented by Ouchi et al. (2015), and Ouchi+ 15 is the ALL-Cases Joint Model in Ouchi et al. (2015). The mark † denotes the significantly better results with the significance level $p < 0.05$, comparing Single-Seq and Multi-Seq.

6.2 Results

Neural Models vs Baseline Models

Table 1 presents F-measures from our neural sequence models with eight RNN or Grid layers and the baseline models on the test set. For the significant test, we used the bootstrap resampling method. According to all metrics, both the single- (Single-Seq) and multi-sequence models (Multi-Seq) outperformed the baseline models. This confirms that our neural models realize high performance, even without syntactic information, by learning contextual information effective for PAS analysis from the word sequence of the sentence.

In particular, for zero arguments (*Zero*), our models achieved a considerable improvement compared to the joint model in Ouchi et al. (2015). Specifically, the single-sequence model improved by approximately 2.0 points, and the multi-sequence model by approximately 3.0 points according to the F-measure. These results suggest that modeling the context of the entire sentence using RNNs are beneficial to Japanese PAS analysis, particularly to zero argument identification.

Effects of Multiple Predicate Consideration

As Table 1 shows, the multi-sequence model significantly outperformed the single-sequence model in terms of the F-measure overall (81.42% vs 81.15%). These results demonstrate that the grid-type neural architecture can effectively capture multi-predicate interactions by connecting the sequences of the argument candidates for all predicates in a sentence.

Compared to the single-sequence model for dif-

L		Single-Seq		Multi-Seq	
		$+res.$	$-res.$	$+res.$	$-res.$
2	*Dep*	**87.34**	87.10	87.43	**87.73**
	Zero	**47.98**	47.90	**47.66**	46.93
	All	**80.62**	80.24	**80.71**	80.68
4	*Dep*	87.27	**87.41**	**87.60**	87.09
	Zero	50.43	**50.83**	48.10	**48.58**
	All	80.92	**80.99**	**80.99**	80.59
6	*Dep*	**87.73**	87.11	**88.04**	87.39
	Zero	48.81	**49.51**	**48.98**	48.91
	All	**81.05**	80.63	**81.19**	80.68
8	*Dep*	**87.98**	87.23	**87.65**	87.07
	Zero	47.40	**48.38**	**49.34**	48.23
	All	**81.31**	80.33	**81.33**	80.40

Table 2: Performance comparison for different numbers of layers on the development set in F-measures. L is the number of the RNN or Grid layers. $+res.$ or $-res.$ indicates whether the model has residual connections ($+$) or not ($-$).

M	Type	No. Args	Single-Seq	Multi-Seq
1	*Dep*	2,733	**89.97**	89.66
	Zero	154	47.62	**53.54**
	All	2,887	**88.08**	88.01
2	*Dep*	5,674	89.64	**90.11**
	Zero	836	53.87	**54.21**
	All	6,510	85.39	**85.95**
3	*Dep*	6,067	87.72	**88.06**
	Zero	1,357	49.98	**51.82**
	All	7,424	81.43	**82.11**
4	*Dep*	4,616	87.80	**87.84**
	Zero	1,205	47.27	**48.50**
	All	5,821	80.31	**80.69**
5+	*Dep*	6.983	**86.63**	86.30
	Zero	2,467	39.83	**40.66**
	All	9,450	**76.17**	76.00

Table 3: Performance comparison for different numbers (M) of predicates in a sentence on the test set in F-measures.

ferent argument types, the multi-sequence model achieved slightly better results for direct dependency arguments (*Dep*) (88.10% vs 88.17%). In addition, for zero arguments (*Zero*), which have no syntactic dependency on their predicate, the multi-sequence model outperformed the single-sequence model by approximately 1.0 point according to the F-measure (46.10% vs 47.12%). This shows that capturing multi-predicate interactions is particularly effective for zero arguments, which is consistent with the results in Ouchi et al. (2015).

Effects of Network Depth

Table 2 presents F-measures from the neural sequence models with different network depths and with/without residual connections. The performance tends to improve as the RNN or Grid layers get deeper with residual connections. In particular, the two models with eight layers and residual connections achieved considerable improvements of approximately 1.0 point according to the F-measure compared to models without residual connections. This means that residual connections contribute to effective parameter learning of deeper models.

Effects of the Number of Predicates

Table 3 presents F-measures from the neural sequence models with different numbers of predicates in a sentence. In Table 3, M denotes how many predicates appear in a sentence. For example, the sentence in Figure 1 includes two predicates, "arrested" and "escaped", and thus in this example $M = 2$.

Overall, performance of both models gradually deteriorated as the number of predicates in a sentence increased, because sentences that contain many predicates are complex and difficult to analyze. However, compared to the single-sequence model, the multi-sequence model suppressed performance degradation, especially for zero arguments ($Zero$). By contrast, for direct dependency arguments (Dep), both models either achieved almost equivalent performance or the single-sequence model outperformed the multi-sequence model. A Detailed investigation of the relation between the number of predicates in a sentence and the complexity of PAS analysis is an interesting line for future work.

Comparison per Case Role

Table 4 shows F-measures for each case role. For reference, we show the results of the previous studies using the NAIST Text Corpus 1.4β with external resources as well.[7]

[7]The major difference between the NAIST Text Corpus 1.4β and 1.5 is the revision of the annotation criterion for the dative case (DAT) (corresponding to Japanese case marker "に"). Argument and adjunct usages of the case marker "に" are not distinguished in 1.4β, making the identification of the dative case seemingly easy (Ouchi et al., 2015).

	Dep			*Zero*		
	NOM	ACC	DAT	NOM	ACC	DAT
NAIST Text Corpus 1.5						
Imamura+ 09	86.50	92.84	30.97	45.56	21.38	0.83
Ouchi+ 15	88.13	92.74	38.39	48.11	24.43	4.80
Single-Seq	88.32	93.89	65.91	49.51	35.07	9.83
Multi-Seq	88.75	93.68	64.38	50.65	32.35	7.52
NAIST Text Corpus 1.4β						
Taira+ 08*	75.53	88.20	89.51	30.15	11.41	3.66
Imamura+ 09*	87.0	93.9	80.8	50.0	30.8	0.0
Sasano+ 11*	-	-	-	39.5	17.5	8.9

Table 4: Performance comparison for different case roles on the test set in F-measures. NOM, ACC or DAT is the nominal, accusative or dative case, respectively. The asterisk (*) indicates that the model uses external resources.

Comparing the models using the NAIST Text Corpus 1.5, the single- and multi-sequence models outperformed the baseline models according to all metrics. In particular, for the dative case, the two neural models achieved much higher results, by approximately 30 points. This suggests that although dative arguments appear infrequently compared with the other two case arguments, the neural models can learn them robustly.

In addition, for zero arguments (*Zero*), the neural models achieved better results than the baseline models. In particular, for zero arguments of the nominative case (NOM), the multi-sequence model demonstrated a considerable improvement of approximately 2.5 points according to the F-measure compared with the joint model in Ouchi et al. (2015). To achieve high accuracy for the analysis of such zero arguments, it is necessary to capture long distance dependencies (Iida et al., 2005; Sasano and Kurohashi, 2011; Iida et al., 2015). Therefore, the improvements of the results suggest that the neural models effectively capture long distance dependencies using RNNs that can encode the context of the entire sentence.

7 Conclusion

In this work, we introduced neural sequence models that automatically induce effective feature representations from the word sequence information of a sentence for Japanese PAS analysis. The experiments on the NAIST Text Corpus demonstrated that the models realize high performance without the need for syntactic information. In particular, our multi-sequence model improved the performance of *zero argument* identification, one of the problematic issues facing Japanese PAS analysis, by considering the *multi-predicate interactions* with Grid-RNNs.

Because our neural models are applicable to SRL, applying our models for multilingual SRL tasks presents an interesting future research direction. In addition, in this work, the model parameters were learned without any external resources. In future work, we plan to explore effective methods for exploiting large-scale unlabeled data to learn the neural models.

Acknowledgments

This work was partially supported by JST CREST Grant Number JPMJCR1513 and JSPS KAKENHI Grant Number 15K16053. We are grateful to the members of the NAIST Computational Linguistics Laboratory and the anonymous reviewers for their insightful comments.

References

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*. pages 152–164.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder

for statistical machine translation. In *Proceedings of EMNLP*. pages 1724–1734.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* .

Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Proceedings of Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop*.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional LSTM networks for improved phoneme classification and recognition. In *Proceedings of International Conference on Artificial Neural Networks*. pages 799–804.

Alex Graves and Jürgen Schmidhuber. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In *Proceedings of NIPS*. pages 545–552.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2013. Japanese zero reference resolution considering exophora and author/reader mentions. In *Proceedings of EMNLP*. pages 924–934.

Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2011. Japanese predicate argument structure analysis exploiting argument position and type. In *Proceedings of IJCNLP*. pages 201–209.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385* .

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2005. Anaphora resolution by antecedent identification followed by anaphoricity determination. *ACM Transactions on Asian Language Information Processing (TALIP)* 4(4):417–434.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the Linguistic Annotation Workshop*. pages 132–139.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ILP solution to zero anaphora resolution. In *Proceedings of ACL-HLT*. pages 804–813.

Ryu Iida, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, and Julien Kloetzer. 2015. Intra-sentential zero anaphora resolution using subject sharing recognition. In *Proceedings of EMNLP*. pages 2179–2189.

Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh, Canasai Kruengkrai, and Julien Kloetzer. 2016. Intra-sentential subject zero anaphora resolution using multi-column convolutional neural network. In *Proceedings of EMNLP*. pages 1244–1254.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of ACL-IJCNLP*. pages 85–88.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2016. Grid long short-term memory. In *Proceedings of ICLR*.

D.P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

Hiroki Ouchi, Hiroyuki Shindo, Kevin Duh, and Yuji Matsumoto. 2015. Joint case argument identification for Japanese predicate argument structure analysis. In *Proceedings of ACL-IJCNLP*. pages 961–970.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML*.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of IJCNLP*. pages 758–766.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* pages 2673–2681.

Tomohide Shibata, Daisuke Kawahara, and Sadao Kurohashi. 2016. Neural network-based model for Japanese predicate argument structure analysis. In *Proceedings of ACL*. pages 1235–1244.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proceedings of EMNLP*. pages 523–532.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Haitong Yang and Chengqing Zong. 2014. Multi-predicate semantic role labeling. In *Proceedings of EMNLP*. pages 363–373.

Katsumasa Yoshikawa, Masayuki Asahara, and Yuji Matsumoto. 2011. Jointly extracting Japanese predicate-argument relation with markov logic. In *Proceedings of IJCNLP*. pages 1125–1133.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL-IJCNLP*.