Performance of Structural Generalization in Connectionist Japanese Complex Sentence Parser

Minoru MOTOKI
Kyushu Sangyo University
2-3-1 Matsukadai, Higashi-ku,
Fukuoka-shi, 813-8503 Japan
motoki@te.kyusan-u.ac.jp

Yoshio SHIMAZU
Kyushu Sangyo University
2-3-1 Matsukadai, Higashi-ku,
Fukuoka-shi, 813-8503 Japan
shimazu@te.kyusan-u.ac.jp

Abstract

We propose the connectionist Japanese complex sentence parser JSPAC. JSPAC is composed of three modules, the simple sentence parser, the segmenter and the stack. The segmenter and the stack segment a complex sentence into simple sentences. The simple sentence parser generates a case structure from the segmented simple sentence. After training JSPAC with the learning data which we generated, we evaluate the JSPAC correctness, especially the ability of the structural generalization for unrestricted recursive structures.

1 Introduction

SPEC proposed by Miikkulainen,R. (Miikkulainen, 1996) is a subsymbolic parsing model which has ability of structural generalization for embedded structures. The basic idea of SPEC is to separate the tasks of segmenting an input word sequence into clauses, forming the case-role representations, and keeping track of the recursive embeddings into the other modules. Each module is trained with only the most basic relative clause constructs, and the combined system is able to generalize to novel sentences with remarkably complex structure. It is important that SPEC is not a neural network reimplementation of a symbol processer.

Jain,A.N. proposed the modular network PARSEC (Jain, 1992) which processed spoken English. PARSEC is a connectionist model and assigns case-roles to English sentences. PARSEC can deal with not only embedded sentences but also passive and conditional constructions. Moreover, PARSEC has robustness against noise. It is interesting that PARSEC was installed in the JANUS system which is a translator, and was shown to be able to process the multi-modal informations such as intonations of dialogues.

However, their models are the systems for English language that has the property of fixed word orders. No system is proposed for Japanese language which has the property that word orders are able to be changed freely, and that is a SOV type language in which verbs locate at the last of sentences. For German language, a connectionist approach was presented as a language acquiring model by Westermann,G. *et al* (Westermann and Miikkulainen, 1994). It is the model for simple sentences.

In this paper, we propose JSPAC(Japanese Subsymbolic PArser for Complex sentence) shown in Figure 1. The JSPAC architecture became similar to SPEC eventually. We experimented the learning and the execution of Japanese sentence parsing by JSPAC, and we evaluated its ability of structural generalization.

2 Task and Basic Principles

The task of JSPAC is to perform the case structure analysis of each clause in a complex sentence. In this paper, we adopted eight case-roles such as AGENT, OBJECT, GOAL, LOCATION, INSTRUMENT for nouns, and ACTION, ACTION(PAST), ACTION(PROGRESSIVE) for verbs.

The proposed model has the ability of structural generalization. It can parse more than two levels embedded sentences by being trained with only one level embedded sentences. For example, being trained with only「庭で遊んでいる子供に犬が吠える。」and「子供に吠える犬を妻が叱った。」, the model can parse「庭で遊んでいる子供に吠える犬を妻が叱った。」.

Although the JSPAC architecture is similar to the SPEC architecture, the executions are different from each other. Japanese language is different from English language as follows.

(a1) As the triggers for moving into embedded clauses, English language has the relative pronoun 'who', while Japanese language has no relative pronoun. The following sentences have the same meaning.

- The wife scolded the dog <u>which</u> was barking at the child.
- 子供に吠えている犬を妻が叱った。(*in Japanese*)

(a2) Japanese language has the property that word orders are able to be changed freely. For exam-

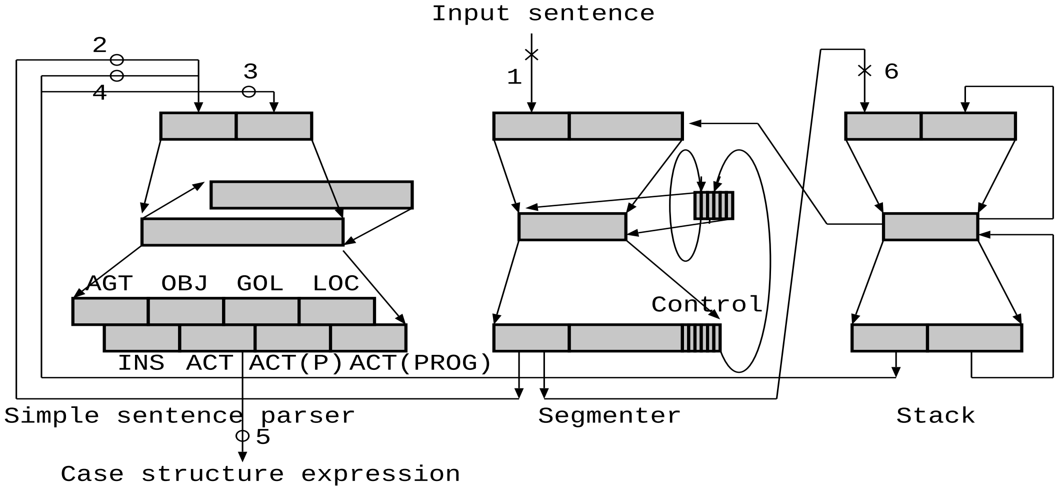

Figure 1: JSPAC Architecture

ple, the following sentence has the same meaning to the above sentence .

- 妻が子供に吠えている犬を叱った。

(a3) Verbs locate at the last of sentences in Japanese language.

Therefore, we resolved these problems with following methods in JSPAC. These (b1) ~ (b3) correspond respectively to (a1) ~ (a3).

(b1) Use verbs as the triggers for moving into embedded clauses.

(b2) Make the input into the simple sentence parser the pair of a noun and a case-particle, because in Japanese language, the word order is not effective and the function of case-particles (or postpositions) is most effective for the case-role assignment. We call this method **Paring-Input**.

(b3) The simple sentence parser is an Elman type network. The inputs into the simple sentence parser is in reverse order of clauses in case of Japanese language. Namely, the verb is given first into the simple sentence parser also in Japanese language.

3 Network Architecture

JSPAC consists of three main modules, the simple sentence parser, the segmenter, and the stack. On executing, the modules form a network of networks, and each of them feeds its output to the input of another module. JSPAC detects verbs to recognize the existence of a clause and see the first independent word following the verb for judging whether that clause is center-embedded or not.

JSPAC uses the distributed representations for words. The word representation is 12 dimension vector whose elements are [0.0, 1.0]. We call this vector **Concept-Vector**. In this paper, they are determined randomly and fixed.

3.1 Simple Sentence Parser

One of the main modules of JSPAC is the simple sentence parser (shown in Figure 2). Its task is the assignment of the case-roles to the independent words of input simple sentences. Eight case-roles are adopted. They are described in section 2.

For dealing with the broad range of sentence length, the network is constructed by an Elman type neural network. The input layer has 24 neural units (12 units $\times$ 2 slots) for two concept-vectors which correspond to two words of an input phrase. This 'phrase' means a pair of one independent word (IND) and one dependent word (DEP) . The output layer has 96 units (12 units $\times$ 8 slots). If the network has successfully learned the task, each concept-vector of the input independent words fills the correct case slot in the output layer.

In Japanese language, the verb phrase that governs the meaning of a sentence is located in the last part of the simple sentence. Naturally, verbs should be given to the simple sentence parser at an early stage. Therefore, we adopt the reverse input method, in which the simple sentence parser is given constituent phrases in the reverse word order.

We also adopt the training strategy called **Keeping-Slot Learning** (Table 1). If the input sentence has three independent words, the learning process takes three back-propagation steps, and at each step the target of the preceding words are kept while the learning process of a sentence continues.

Table 1: Keeping-Slot Learning

step	IND	DEP	AGT	OBJ	GOL	LOC	INS	ACT	ACT(P)	ACT(PROG)
1	遊ん	でいる	NULL	NULL	NULL	NULL	NULL	NULL	NULL	遊ぶ
2	庭	で	NULL	NULL	NULL	庭	NULL	NULL	NULL	遊ぶ
3	子供	が	子供	NULL	NULL	庭	NULL	NULL	NULL	遊ぶ

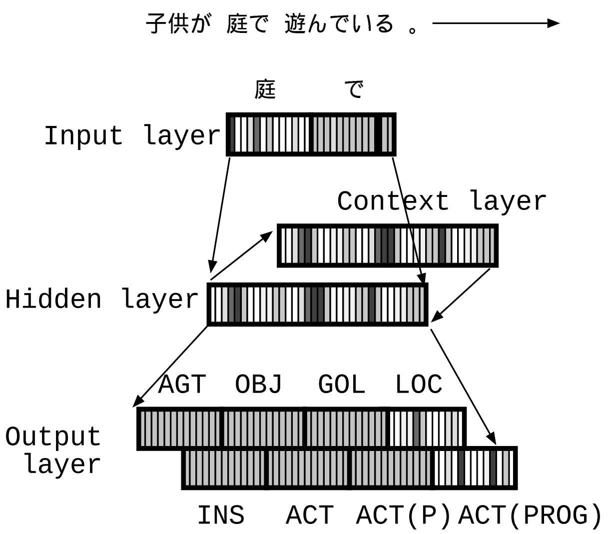

Figure 2: Simple Sentence Parser

3.2 Stack

The stack is constructed by a RAAM(Recursive Auto Associative Memory) network. The stack acts as a stack, a kind of computer data structures. Basically, a sequence of words constructing a input complex sentence are pushed into the stack. The tasks of the stack in JSPAC are as follows:

(1) Give the segmenter the activatied pattern of the stack hidden layer, so that the segmenter can consider all of the preceding words of an input complex sentence.

(2) Realize the reverse order input of phrases constrcuting a simple sentence by popping from the stack, when the simple sentence parser is given input vectors.

3.3 Segmenter

The segmenter is constructed by an auto-associative network and a Jordan network which can learn time series. The segmenter works as a governor of all modules. The concept-vector of the next input word and the activated pattern of the stack hidden layer are given to the input layer of the segmenter. The output layer of the segmenter includes the units, the number of which is equals the number of the input layer units, and 6 units for the control signals. This ‘control signals’ means the signals that control the information transfers among the modules.

The segmenter is trained so that on its output layer, it can give the same activation pattern as the input, and the control signals generated in order to make the whole network of networks execute its task in an appropriate time sequence. The main tasks of the segmenter are to detect verbs from the input word sequence and to generate the control signals for segmenting a complex sentence into simple sentences (or clauses).

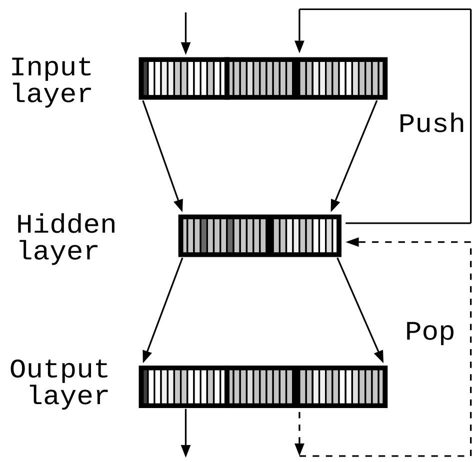

Figure 3: Stack

3.4 Control

The segmenter generates the control signals to govern the whole network of networks. These contorl signals consist of excitatory signals and inhibitatory signals. The excitatory control signals open the gates when the values of those signals are 1. On the other hand, the inhibitatory control signals open the gates when the values of those signals are 0. In Figure 1, the gate controled by the excitatory control signal is shown with a simbol ‘○’, while the gate controled by the inhibitatory control signal is shown with a simbol ‘×’. There are two inhibitatory control signals, unit 1 and unit 6. As an example, we show in Table 2, the controls signals generated when

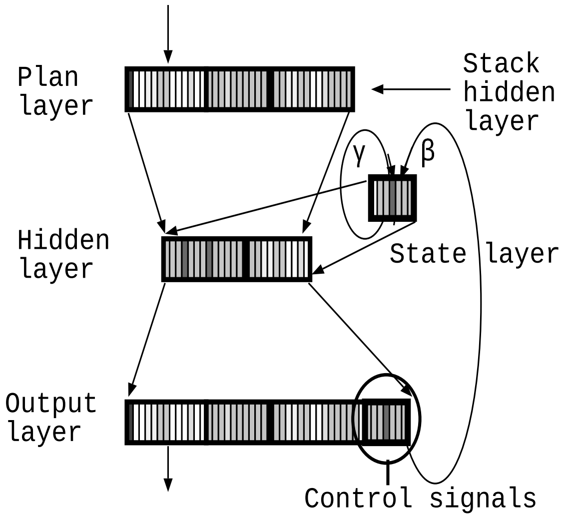

Figure 4: Segmenter

Table 2: Control Signals

Input words	Control signals 1 2 3 4 5 6	
庭	0 0 0 0 0 0	
で	0 0 0 0 0 0	
遊ん	0 0 0 0 0 0	
でいる	0 0 0 0 0 0	
子供	1 0 1 0 0 1	
子供	1 0 0 1 0 1	
子供	1 0 1 0 0 1	
子供	1 0 0 1 0 1	
子供	0 1 0 0 1 0	Output of a case structure
に	0 0 0 0 0 0	
犬	0 0 0 0 0 0	
が	0 0 0 0 0 0	
吠える	0 0 0 0 0 0	
。	1 0 0 1 0 1	
。	1 0 1 0 0 1	
。	1 0 0 1 0 1	
。	1 0 1 0 0 1	
。	1 0 0 1 1 1	Output of a case structure

JSPAC are processing a Japanese complex sentence 「庭で遊んでいる子供に犬が吠える。」.

3.5 Execution Process

In this section, we explain the execution process of JSPAC. We now consider the case where the following Japanese complex sentence is given.

- 庭で遊んでいる子供に犬が吠える。

The task of JSPAC is to assign the following cases respectively to the subordinate clause and the principal clause.

- 庭で / LOC   子供が / AGT   遊んでいる / ACT(PROG)
- 子供に / OBJ   犬が / AGT   吠える / ACT

We now explain Figure 5 step by step.

(1) '庭' is given to the segmneter. And '庭' are pushed from the segmenter output layer into the stack input layer (Figure 5-(1)). The following words are similarly pushed into the stack .

(2) The segmenter detect the verb '遊ん'. And '遊ぶ' is pushed from the segmenter output layer into the stack (Figure 5-(2)).

(3) Since the next word of the verb '遊ん' is the auxiliary verb 'でいる', 'ている' are pushed into the stack. Then, 'ている' are popped from the stack and transmitted to the input slot for a dependent word in the simple sentence parser (Figure 5-(3)).

(4) When the modified word '子供' is given to the segmenter, JSPAC must stop being given the following words. Then, the auxiliay verb 'ている' is popped from the stack and transmitted to the input slot for a dependent word in the simple sentence parser. And then, the verb '遊ぶ' is popped from the stack and transmitted to the input slot for an independent word in the simple sentence parser. Afterwards, the simple sentence parser throw the verb '遊ぶ' into the case slot ACT(PROG)(Figure 5-(4)).

(5) Next, 'で' stored in the stack is popped and transmitted to the input slot for a dependent word in the simple sentence parser. Then, '庭' stored in the stack is popped and transmitted to the input slot for an independent word in the simple sentence parser. '庭' is thrown into the case slot LOC. Then, '子供' is transmitted from the segmenter output into the input slot for an independent word in the simple sentence parser. At the same time, '子供' in the segmenter output is pushed into the stack. '子供' is thrown into the case slot AGT. And then, the case structure expression of the subordinate clause is picked up from the output layer (Figure 5-(5)). The context layer of the simple sentence parser is cleared. Afterwards, the analysis of the principal clause begins.

(6) 'に' is given to the segmenter and is pushed from the segmenter output into the stack (Figure 5-(6)).

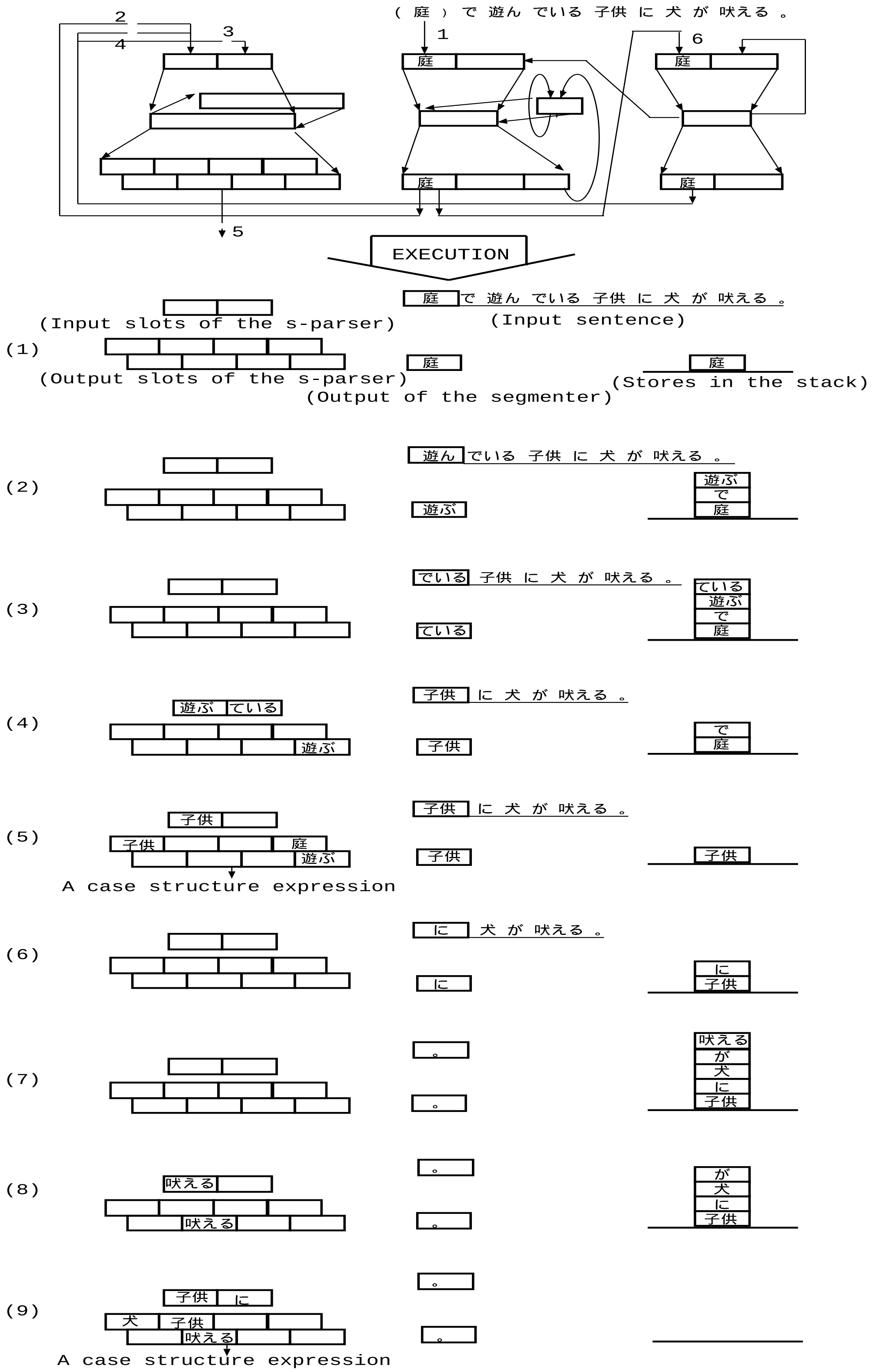

Figure 5: Execution Process

Table 3: Words Used in the Experiment

Noun(12)	妻 (wife) 子供 (child) 犬 (dog) 猫 (cat) 雪 (snow) 雨 (rain) 雪割草 (hepetica) 木 (tree) 塀 (fence) 庭 (garden) 花壇 (flower bed) 犬小屋 (kennel)
Verb(8)	登る (climb) 入る (enter) 走る (run) 遊ぶ (play) 降る (fall) 植える (plant) 吠える (bark) 叱る (scold)
Post Position(4)	が (*ga*) を (*wo*) で (*de*) に (*ni*)
Auxiliary Verb(2)	た (*ta*) ている (*teiru*)
Other(2)	。(.) NULL

(7) The following words '犬', 'が', '吠える' are similarly pushed into the stack. Next, the period '。' is given to the segmenter (Figure 5-(7)).

(8) When the segmenter detects the period '。', '吠える' is popped from the stack and transmitted to the simple sentence parser. '吠える' is thrown into the case slot ACT (Figure 5-(8)).

(9) Next, 'が' and '犬' are popped from the stack, transmitted to the simple sentence parser, and '犬' is thrown into the case slot AGT. Then, 'に' and '子供' are popped from the stack, transmitted to the simple sentence parser, and '子供' is thrown into the case slot OBJ. Finally, the case structure expression of the principal clause is picked up from the output layer, (Figure 5-(9)), and the context layer of the simple sentence parser are cleared).

4 Experiment

4.1 Data

Twenty six (26) words used in the computational experiment are shown in Table 3. We generated 952 complex sentences using these words as described below (see Figure 6).

First, we generated the simple sentences which are semantically proper, or satisty the selectional restriction of the verbs as shown in Table 4, and constructed a small world of Japanese language by describing various situations which are made by use of the words, '雨', '雪', '犬', '猫', '妻' and '子供' in someone's garden. We used these simple sentences as clause data.

Next, we generated the complex sentences to use these clause data automatically with computer programs according to the following procedure.

(1) Prepare a set of simple sentence templates T0 by changing the order of phrases in the sentences of these clause data (Table 4).

Table 4: Clause Data

No.	Sentence templates	Case-roles
1	子供 が 塀 に 登る	AGT GOL ACT
2	子供 が 木 に 登る	AGT GOL ACT
3	猫 が 塀 に 登る	AGT GOL ACT
4	猫 が 木 に 登る	AGT GOL ACT
5	犬 が 犬小屋 に 入る	AGT GOL ACT
6	猫 が 犬小屋 に 入る	AGT GOL ACT
7	子供 が 庭 を 走る	AGT LOC ACT
8	犬 が 庭 を 走る	AGT LOC ACT
9	猫 が 庭 を 走る	AGT LOC ACT
10	子供 が 庭 で 遊ぶ	AGT LOC ACT
11	子供 が 雪 で 遊ぶ	AGT INS ACT
12	雪 が 庭 に 降る	AGT GOL ACT
13	雨 が 庭 に 降る	AGT GOL ACT
14	雪 が 木 に 降る	AGT GOL ACT
15	雪 が 犬小屋 に 降る	AGT GOL ACT
16	妻 が 雪割草 を 花壇 に 植える	AGT OBJ GOL ACT
17	子供 が 雪割草 を 花壇 に 植える	AGT OBJ GOL ACT
18	犬 が 子供 に 吠える	AGT OBJ ACT
19	犬 が 猫 に 吠える	AGT OBJ ACT
20	妻 が 子供 を 叱る	AGT OBJ ACT
21	妻 が 犬 を 叱る	AGT OBJ ACT

(2) Obtain a set of one level embedded sentence templates T1, by embedding the clause data into the sentences of T0 if a common noun is found in both the clause data and the sentences of T0.

(3) Obtain a set of two levels embedded sentence templates T2, by embedding the clause data into the sentence of T1 if a common noun is found in both the clause data and the sentences of T1.

(4) Remove the sentences in which the modified words act AGT roles in both the principal clause and the subordinate clause in T1 and T2.

(5) Remove the sentences which contain more than two same nouns in T1 and T2.

Finally, we added auxiliary verbs to the sentence templates in T1 and T2, and selected natural complex sentences according to the language sense of Japanese native speakers.

As a result, we obtained 952 sentences. Those were 50 simple sentences, 503 one level embedded sentences and 399 two levels embedded sentences. In this experiment, we generated only sentences, the tense of whose principal clause is present, and did not generate sentences, the tense of whose principal clause is past or progressive. Now, we show the sentence patterns of the generated sentences in Table 5.

Table 5: Patterns of Generated Sentences

No.	Total	Sentence Pattern
1	38	N PP N PP V 。
2	12	N PP N PP N PP V 。
3	15	(N PP V) N PP N PP V 。
4	118	(N PP V AUX) N PP N PP V 。
5	4	(N PP V AUX) N PP N PP N PP V 。
6	2	(N PP N PP V AUX) N PP N PP V 。
7	15	N PP (N PP V) N PP V 。
8	119	N PP (N PP V AUX) N PP V 。
9	4	N PP (N PP V AUX) N PP N PP V 。
10	2	N PP (N PP N PP V AUX) N PP V 。
11	4	N PP N PP (N PP V AUX) N PP V 。
12	23	(N PP V AUX) N PP (N PP V) N PP V 。
13	24	(N PP V) N PP (N PP V AUX) N PP V 。
14	171	(N PP V AUX) N PP (N PP V AUX) N PP V 。
15	1	(N PP V AUX) N PP (N PP N PP V AUX) N PP V 。
16	1	(N PP N PP V AUX) N PP (N PP V AUX) N PP V 。
17	1	((N PP V) N PP V) N PP N PP V 。
18	36	((N PP V AUX) N PP V) N PP N PP V 。
19	39	((N PP V) N PP V AUX) N PP N PP V 。
20	32	((N PP V AUX) N PP V AUX) N PP N PP V 。
21	2	((N PP V AUX) N PP V AUX) N PP N PP N PP V 。
22	1	((N PP N PP V) N PP V AUX) N PP N PP V 。
23	1	((N PP N PP V AUX) N PP V) N PP N PP V 。
24	43	N PP ((N PP V AUX) N PP V) N PP V 。
25	23	N PP ((N PP V) N PP V AUX) N PP V 。
26	46	N PP ((N PP V AUX) N PP V AUX) N PP V 。
27	2	N PP ((N PP V AUX) N PP V AUX) N PP N PP V 。
28	1	N PP ((N PP N PP V) N PP V AUX) N PP V 。
29	1	N PP ((N PP N PP V AUX) N PP V) N PP V 。
30	2	N PP N PP ((N PP V AUX) N PP V AUX) N PP V 。
31	2	((N PP V) N PP V AUX) N PP (N PP V) N PP V 。
32	6	((N PP V AUX) N PP V AUX) N PP (N PP V) N PP V 。
33	27	((N PP V AUX) N PP V) N PP (N PP V AUX) N PP V 。
34	11	((N PP V) N PP V AUX) N PP (N PP V AUX) N PP V 。
35	38	((N PP V AUX) N PP V AUX) N PP (N PP V AUX) N PP V 。
36	1	(N PP V) N PP ((N PP V) N PP V AUX) N PP V 。
37	28	(N PP V AUX) N PP ((N PP V AUX) N PP V) N PP V 。
38	20	(N PP V AUX) N PP ((N PP V) N PP V AUX) N PP V 。
39	2	(N PP V) N PP ((N PP V AUX) N PP V AUX) N PP V 。
40	21	(N PP V AUX) N PP ((N PP V AUX) N PP V AUX) N PP V 。
41	4	((N PP V AUX) N PP V AUX) N PP ((N PP V AUX) N PP V) N PP V 。
42	1	((N PP V AUX) N PP V AUX) N PP ((N PP V) N PP V AUX) N PP V 。
43	4	((N PP V AUX) N PP V) N PP ((N PP V AUX) N PP V AUX) N PP V 。
44	1	((N PP V) N PP V AUX) N PP ((N PP V AUX) N PP V AUX) N PP V 。
45	3	((N PP V AUX) N PP V AUX) N PP ((N PP V AUX) N PP V AUX) N PP V 。

N:Noun V:Verb PP:PostPosition(case-particle) AUX: AUXiliary verb

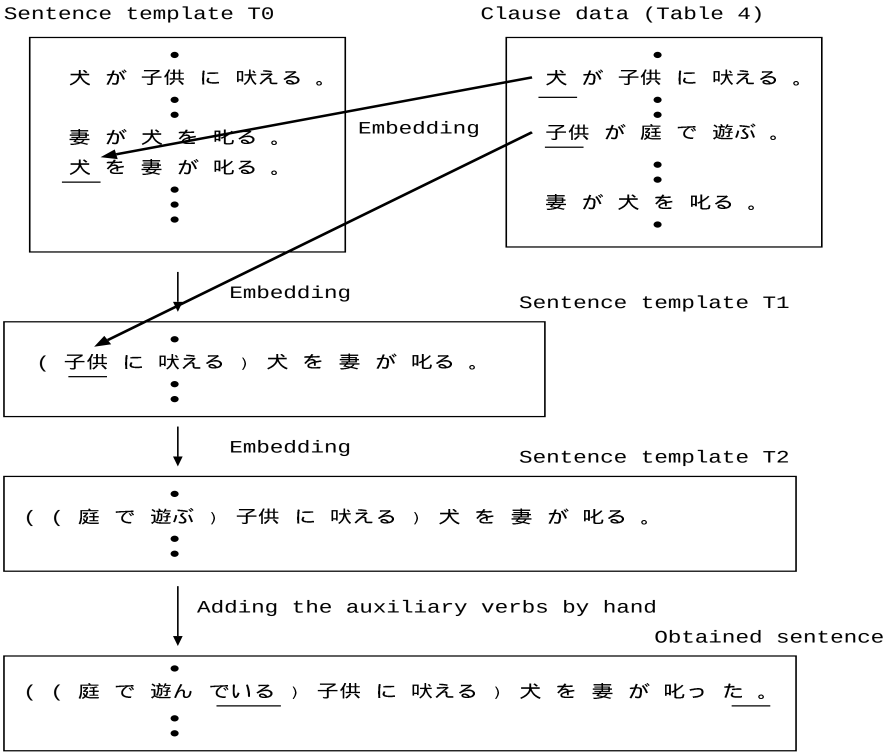

Figure 6: Generating the Experiment Data

4.2 Training

For training, we selected randomly 353 one level embedded sentences (70% of all). Remained sentences (simple sentences, 30% of one level embedded sentences, and 70% of two levels embedded sentences) were used as test data.

There are a variety of strategies for training a modular system such as JSPAC. One possibility is to train all module of the entire system as a whole, while another possibility is to train each module separately. In this experiment, we adopted the former strategy, but strictly speaking, slightly different. While training, the correct control signals were used for gating operations, instead of the values containing the error of unfinished training. Therefore, the concept-vectors were transmitted exactly from one module to another module.

The network parameters of each module were as follows. The number of both hidden units and the context units of the simple sentence parser was 90. The number of hidden units of both the segmenter and the stack was 60. The number of concept-vector elements were 12. Hence, the number of input units of both the segmenter and the stack was 72 (60 hidden units + 12 concept-vector elements). The sigmoidal gradients of all modules were set to 1.0.

We adopted the pure error back-propagation algorithm (Rumellhart and Williams, 1986) without the momentum term, backward-propageted immediately after forward-propageted for each input. The learning rate of all modules were fixed to 0.1. In the Jordan network, γ was set to 0.5 and β was set to 0.5. The initial values of the network link-weights and biases were set to random values in the range of [-1.0,+1.0].

We stopped training when the average RMS(Root Mean Square) error per output unit of the simple sentence parser reached at 0.015. At that time, the number of the training epochs was 654, and the average RMS error per output unit of the segmenter and the stack were 0.0263 and 0.0164, respectively. We show the training curve in Figure 7.

We set the executing conditions as follows. In order to identify the words at the simple sentence parser output, the Euclidean distance was used, and the words which are closest to the output were searched among all of the input words. The control signal in the segmenter output was recognized as 1 when the unit output value was bigger than 0.5, while the control signal was recognized as 0 when the unit output value was equal to or less than 0.5. The correctness of the simple sentence parser output was defined as the case when case structure expressions were picked up in correct timing and all of the words

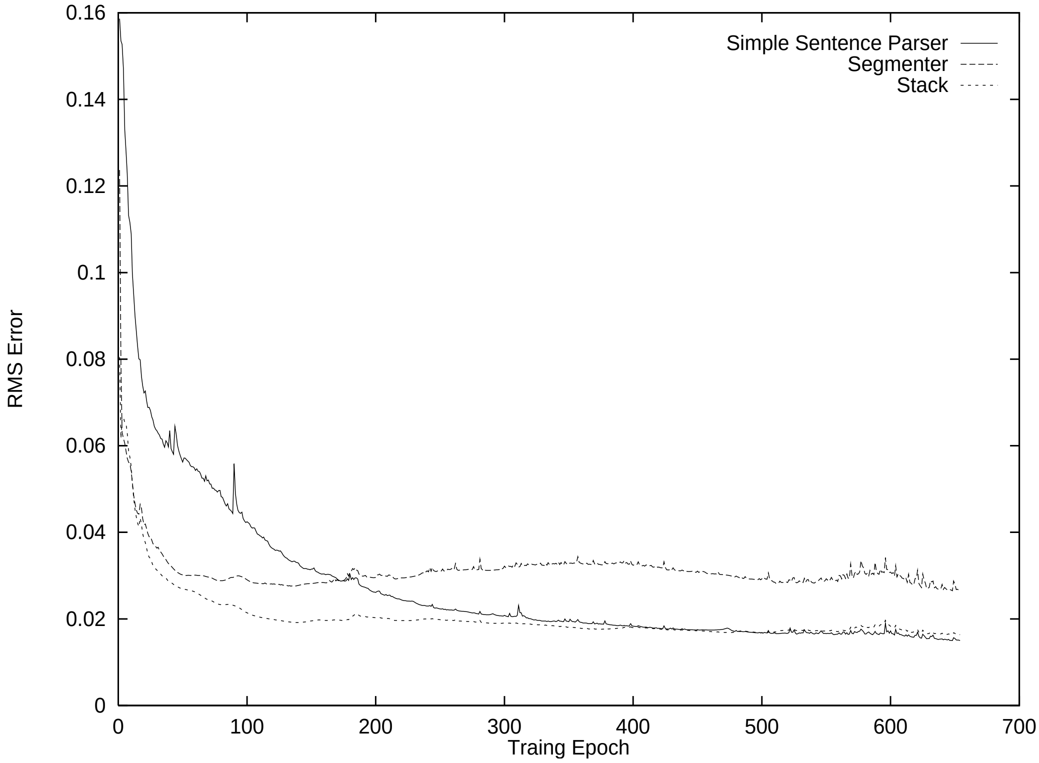

Figure 7: Training Curve

in the expressions were in correct cases.

4.3 Results

4.3.1 Structural Generalization

After training, we examined the JSPAC execution using simple sentences and complex sentences. We show the result in Table 6. Being trained with only one level embedded sentences, the percentage of correct answeres were 79.2% for two levels embedded sentences. Namely, it is proved that JSPAC has ability of structural generalization.

4.3.2 Results on Sentence Patterns

The result on the sentence patterns (Table 5) is shown at Table 7, where the one level embedded sentences 3 ~ 16 include both learned sentences and unlearned sentences. The sentence patterns 24 ~ 30 are the two levels embedded sentences including center-embedded clauses, and have the correctness 60.2% lower than 81.3% of the sentence pattern 17 ~ 23 which are two level embedded sentences but don't have center-embedded clauses.

5 Discussion

There are 83 two levels embedded sentences which are analyzed incorrectly, including 35 sentences in which the segmenter control signals are mistaken and 48 sentences in which the parser outputs are misteken. But it is expected that those sentences are improved in correctness by tuning the network parameters.

As described in section 4.3.2, the two levels embedded sentences with center-embedded clauses are lower than the head-embedded, because it is difficult that the segmenter decides the range of popping from the stack and the stack reproduces the vectors of pushed words.

In this paper, the concept-vectors are determined randomly and fixed. If we use the concept-vectors reflecting the concept of words, the execution performance will be improved. There are FGREP proposed by Miikkulainen,R. and CAN proposed by Takahashi,N. and Motoki,M. (Takahashi and Motoki, 1999) as the systems acquiring the proper vector representations which reflect the concept of words.

When we started to construct a natural language understanding system with connectionist models, we introduced a working hypothesis as follows. Our brain has the whole world image consisting of many small parallel worlds. Therefore, we made a hypothesis of the existence of the **language columns** to express the small worlds. The language column includes all subsystems necessary to language process-

Table 6: Results on Strucures

Structrures	Correct	/	Total	Percent(%)
1-level embeded (learned)	330	/	353	93.5
Simple sentence	29	/	50	58.0
1 level embedded (unlearned)	134	/	150	89.3
2 levels embedded	316	/	399	79.2

Table 7: Results on Sentence Patterns

Sentence Pattern	Correct	/	Total	Percent(%)
1,2	29	/	50	58.0
3 ~ 6	126	/	139	90.6
7 ~ 11	131	/	144	91.0
12 ~ 16	207	/	220	94.0
17 ~ 23	91	/	112	81.3
24 ~ 30	71	/	118	60.2
31 ~ 35	72	/	84	85.7
36 ~ 40	69	/	72	95.8
41 ~ 45	13	/	13	100.0

ing such as the sentence parsing system and the sentence generating system. This paper described the method to construct the parsing system for one of the small worlds.

6 Conclusion

In this paper, we proposed the Japanese complex sentence parser JSPAC, and showed the ability of the structural generalization with the computational experiments. Interestingly, the JSPAC architecture for Japanese processing is similar to the SPEC architecture for English processing. This common network architecture is expected to explain the innateness of the language acquiring ability.

References

A.N. Jain. 1992. Generalization performance in PARSEC—A structured connectionist parsing architecture. In John E. Moody, Stephen J. Hanson, and Richard P. Lippmann, editors, *NIPS4*, pages 209–216. KAUF, KAUF-ADDR.

R. Miikkulainen. 1996. Subsymbolic parsing of embedded structures. In R.Sun and L.A.Bookman, editors, *Computational Architecutures Integrating Neural and Symbolic Processes*, pages 153–186. kluwer academic publisher.

D.E. Rumellhart, G.E. Hinton, and R.J. Williams. 1986. *Learning Internal Representations by Error Propagation*, volume 1. The MIT Press, Massachusetts.

N. Takahashi and M. Motoki. 1999. A subsymbolic approach for acquiring semantic representations. In *Proceedings of the Third International Workshop on Computational Semantics.*

G. Westermann and R. Miikkulainen. 1994. Verb inflections in german child language. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society.*