Chunking and Dependency Analysis of Japanese Compound Functional Expressions by Machine Learning

Takehito Utsuro[1], Takao Shime[2], Masatoshi Tsuchiya[3], Suguru Matsuyoshi[4], and Satoshi Sato[4]

[1] Graduate School of Systems and Information Engineering, University of Tsukuba, 1-1-1, Tennodai, Tsukuba, 305-8573, JAPAN
[2] NEC Corporation, JAPAN
[3] Computer Center, Toyohashi University of Technology, Tenpaku-cho, Toyohashi, 441–8580, JAPAN
[4] Graduate School of Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603, JAPAN

Abstract. This paper proposes an approach of processing Japanese compound functional expressions by identifying them and analyzing their dependency relations through a machine learning technique. The results of experimental evaluation shows that, the dependency analysis model applied to the results of identifying compound functional expressions significantly outperforms the one applied to the results without identifying compound functional expressions.

1 Introduction

In the Japanese language, recognition and semantic interpretation of compound functional expressions are difficult because it often happens that one compound expression may have both a literal (in other words, compositional) *content word* usage and a non-literal (in other words, non-compositional) *functional* usage. For example, Table 1 shows two example sentences of a compound expression "ni tsuite", which consists of a post-positional particle "ni", and a conjugated form "tsuite" of a verb "tsuku". In the sentence (A), the compound expression functions as a case-marking particle and has a non-compositional functional meaning "*about*". On the other hand, in the sentence (B), the expression simply corresponds to a literal concatenation of the usages of the constituents: the post-positional particle "ni" and the verb "tsuite", and has a content word meaning "*follow*". When considering machine translation of those Japanese sentences into English, it is necessary to precisely judge the usage of the compound expression "ni tsuite".

There exist widely-used Japanese text processing tools, i.e., pairs of a morphological analysis tool and a subsequent parsing tool, such as JUMAN[1]+ KNP[2]

[1] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman-e.html
[2] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/knp-e.html

Table 1. Translation Selection of a Japanese Compound Expression "ni tsuite"

(A)	watashi	ha	kare	ni tsuite	hanashita
	(*I*)	(*TOP*)	(*he*)	(*about*)	(*talked*)
	(I talked about him.)				
(B)	watashi	ha	kare	ni tsuite	hashitta
	(*I*)	(*TOP*)	(*he*)	(*ACC*) (*follow*)	(*ran*)
	(I ran following him.)				

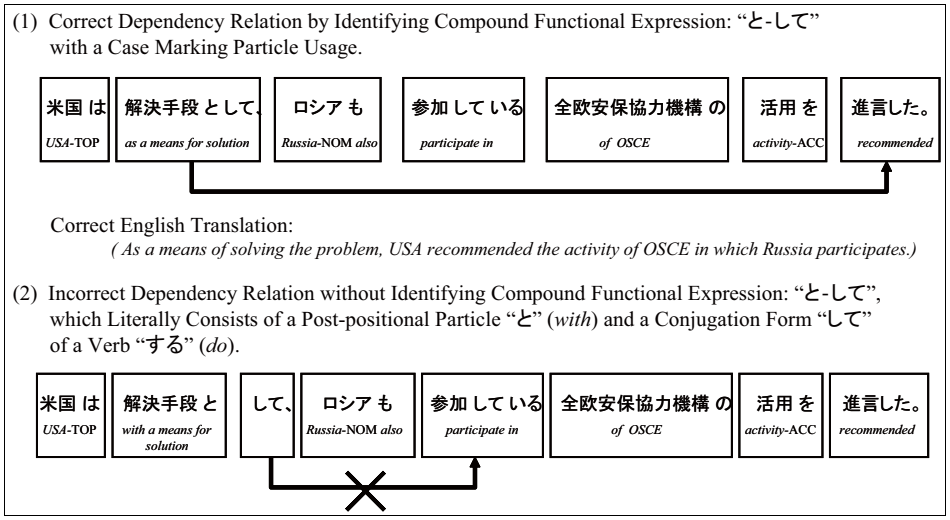

Fig. 1. Example of Improving Dependency Analysis of Compound Functional Expressions by Identifying them before Dependency Analysis

and ChaSen[3]+ CaboCha[4]. However, they process those compound expressions only partially, in that their morphological analysis dictionaries list only limited number of compound expressions. Furthermore, even if certain expressions are listed in a morphological analysis dictionary, those existing tools often fail in resolving the ambiguities of their usages, such as those in Table 1. This is mainly because the framework of those existing tools is not designed so as to resolve such ambiguities of compound (possibly functional) expressions by carefully considering the context of those expressions.

Actually, as a first step of studying computational processing of compound functional expressions, we start with 125 major functional expressions which have non-compositional usages [4], as well as their variants (337 expressions in total). Out of those 337 expressions, 111 have both a *content word* usage and

[3] http://chasen.naist.jp/hiki/ChaSen/
[4] http://chasen.org/~taku/software/cabocha/

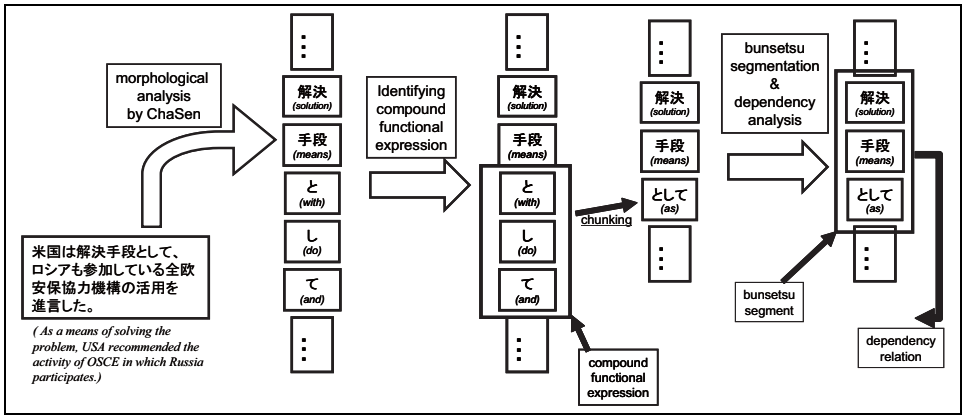

Fig. 2. Overall Flow of Processing Compound Functional Expressions in a Japanese Sentence

a *functional* usage[5]. However, the pair of JUMAN+KNP is capable of distinguishing the two usages only for 43 of the 111 expressions, and the pair of ChaSen+CaboCha only for 40 of those 111 expressions. Furthermore, the failure in distinguishing the two usages may cause errors of syntactic analysis. For example, Figure 1 shows an example of improving dependency analysis of compound functional expressions by identifying them before dependency analysis. Here, Japanese dependency structure is usually defined in terms of the relationship between phrasal units called *bunsetsu* segments (hereafter "bunsetsus"). A bunsetsu consists of one or more content words following zero or any number of functional words. In Figure 1, (1) gives an example of identifying a correct modifiee of the bunsetsu segment including a Japanese compound functional expression "to-shite (*as*)", by appropriately detecting the compound functional expression before dependency analysis. On the other hand, (2) of Figure 1 gives an example of incorrectly indicating an erroneous modifiee of the bunsetsu "shite", which actually happens if we do not identify the compound functional expression "to-shite (*as*)" before dependency analysis of this sentence.

Considering such a situation, it is necessary to develop a tool which properly recognizes and semantically interprets Japanese compound functional expressions. This paper proposes an approach of processing Japanese compound functional expressions by identifying them and analyzing their dependency relations through a machine learning technique. The overall flow of processing compound functional expressions in a Japanese sentence is illustrated in Figure 2. First of all, we assume a sequence of morphemes obtained by a variant of ChaSen with all the compound functional expressions removed from its outputs, as an input to our procedure of identifying compound functional expressions and analyzing

[5] In this paper, we take an approach of regarding each of those variants as a fixed expression, rather than a semi-fixed expression or a syntactically-flexible expression [5].

their dependency relations. We formalize the task of identifying Japanese compound functional expressions in a text as a machine learning based chunking problem [7]. We employ the technique of Support Vector Machines (SVMs) [9] as the machine learning technique, which has been successfully applied to various natural language processing tasks including chunking tasks such as phrase chunking [1] and named entity chunking. Next, against the results of identifying compound functional expressions, we apply the method of dependency analysis based on the cascaded chunking model [2].

2 Chunking Compound Functional Expressions by SVMs

This section describes details of formalizing the chunking task using SVMs. In this paper, we use an SVMs-based chunking tool YamCha[6] [1]. In the SVMs-based chunking framework, SVMs are used as classifiers for assigning labels for representing chunks to each token. In our task of chunking Japanese compound functional expressions, each sentence is represented as a sequence of morphemes, where a morpheme is regarded as a token.

Given a candidate expression, we classify the usages of the expression into two classes: *functional* and *content*. Accordingly, we distinguish the chunks of the two types: the *functional* type chunk and the *content* type chunk. For representing proper chunks, we employ IOB2 representation [1]. As for extending SVMs to multi-class classifiers, we employ the *pairwise* method.

For the feature sets for training/testing of SVMs, we use the information available in the surrounding context, such as the morphemes, their parts-of-speech tags, as well as the chunk labels. More precisely, suppose that we identify the chunk label c_i for the i-th morpheme:

	$\longrightarrow$ Parsing Direction $\longrightarrow$				
Morpheme	m_{i-2}	m_{i-1}	m_i	m_{i+1}	m_{i+2}
Feature set at a position	F_{i-2}	F_{i-1}	F_i	F_{i+1}	F_{i+2}
Chunk label	c_{i-2}	c_{i-1}	$\boxed{c_i}$		

Here, m_i is the morpheme appearing at i-th position, F_i is the feature set at i-th position, and c_i is the chunk label for i-th morpheme. Roughly speaking, when identifying the chunk label c_i for the i-th morpheme, we use the feature sets F_{i-2}, F_{i-1}, F_i, F_{i+1}, F_{i+2} at the positions $i-2$, $i-1$, i, $i+1$, $i+2$, as well as the preceding two chunk labels c_{i-2} and c_{i-1}. The detailed definition of the feature set F_i at i-th position is given in [7], which mainly consists of morphemes as well as information on the candidate compound functional expression at i-th position.

[6] http://chasen.org/~taku/software/yamcha/

3 Coping with Compound Functional Expressions in Statistical Dependency Analysis

In general, dependency relations of a Japanese sentence satisfies the following two constraints:

1. Japanese is a head-final language. Thus, except for the rightmost one, each bunsetsu segment modifies exactly one bunsetsu segment among those appearing to its right.
2. Dependencies do not cross one another.

As the framework of statistical dependency analysis of Japanese sentences, we employ the method of dependency analysis based on the cascaded chunking model [2]. Unlike probabilistic dependency analysis models of Japanese, the cascaded chunking model of [2] does not require the probabilities of dependencies and parses a sentence deterministically.

Bunsetsu Representation As we show in Figure 2, a compound functional expression is identified as a sequence of several morphemes and then chunked into one morpheme. The result of this identification process is then transformed into the sequence of bunsetsu segments. Finally, to this modified sequence of bunsetsu segments, the method of dependency analysis based on the cascaded chunking model is applied. Here, when chunking a sequence of several morphemes constituting a compound functional expression, the following two cases may exist:

(A) As in the case of the example (A) in Table 1, the two morphemes constituting a compound functional expression "ni tsuite" overlaps the boundary of two bunsetsu segments. In such a case, when chunking the two morphemes into one morpheme corresponding to a compound functional expression, those two bunsetsu segments are concatenated into one bunsetsu segment.

kare ni	tsuite	$\Longrightarrow$	kare ni-tsuite
(*he*)			(*he*) (*about*)

(B) As we show below, a compound functional expression "koto ga aru" overlaps the boundary of two bunsetsu segments, though the two bunsetsu segments concatenating into one bunsetsu segment does not include content words. In such a case, its immediate left bunsetsu segment, which corresponds to the content word part of "koto ga aru", has to be concatenated into the one of the compound functional expression "koto ga aru".

itt ta	koto ga	aru	$\Longrightarrow$	itt ta koto-ga-aru
(*went*)				(*have been* $\sim$)

Next, to the compound functional expression, we assign one of the four grammatical function types (case-marking particle, conjunctive particle, adnominal particle, and auxiliary verb types) as its POS tag. For example, the compound functional expression "ni tsuite" in (A) above is assigned the grammatical function type "case-marking particle type", while "koto ga aru" in (B) is assigned "auxiliary verb type".

Table 2. Statistics of Data Sets

	Usages			# of
	functional	content	total	sentences
for chunker training	1918	1165	3083	2429
Kyoto text corpus	5744	1959	7703	38400

Feature Representation As features of dependency analysis, we simply use those of the Japanese dependency analyzer based on the cascaded chunking model, namely, the publicly available version of CaboCha [2]. In CaboCha, for the pair of modifier/modifiee bunsetsu segments, head words and their parts-of-speech tags, inflection-types/forms, functional words and their parts-of-speech tags, inflection-types/forms, inflection forms of the words that appear at the end of bunsetsu segments are used as features. As for features between modifier/modifiee bunsetsu segments, the distance of modifier/modifiee bunsetsu segments, existence of case-particles, brackets, quotation-marks, and punctuation-marks are used.

The modifications of the busetsu representations in the previous section finally cause differences in the feature representations of CaboCha. For example, let us compare the feature representations of the modifier bunsetsu segments in (1) and (2) of Figure 1. In (1), the modifier bunsetsu segment is "kaiketsu-shudan-to-shite (*as a means for solution*)" which has the compound functional expression "to-shite" in its functional word part. On the other hand, in (2), the modifier bunsetsu segment is "shite", which corresponds to the literal verb usage of a part of the compound functional expression "to-shite". In the final feature representations below, this difference causes the following differences in head words, functional words, and POS of the modifier bunsetsu segments:

	(1) of Figure 1	(2) of Figure 1
head word	shudan (*means*)	suru (*do*)
functional word	to-shite (*as*)	te (*and*)
POS	case-marking particle - compound functional expression	conjunctive particle

4 Experimental Evaluation

4.1 Training/Test Data Sets

In the experimental evaluation, we focus on 59 expressions which have balanced distribution of their usages in the newspaper text corpus and are among the most difficult ones in terms of their identification in a text. Here, we assume that we can similarly train chunkers of compound functional expressions other than the

Table 3. Evaluation Results of Chunking Compound Functional Expressions (%)

	Identifying *functional* chunks			Acc. of classifying *functional / content* chunks
	Prec.	Rec.	$F_{\beta=1}$	
majority (= *functional*)	74.6	100	85.5	74.6
Juman/KNP	85.8	40.5	55.0	58.4
ChaSen/CaboCha	85.2	26.7	40.6	51.1
SVM	91.4	94.6	**92.9**	**89.3**

Table 4. Accuracies of Identifying Modifier(s)/Modifiee of Compound Functional Expressions (%)

		modifier	modifiee
baselines	CaboCha (w/o FE)	72.5	**88.0**
	CaboCha (public)	73.9	87.6
chunker + CaboCha (proposed)		**74.0**	**88.0**
reference + CaboCha (proposed)		74.4	88.1

selected 59 expressions. For the training of chunking compound functional expressions, we collected 2,429 example sentences from 1995 Mainichi newspaper text corpus. For the testing of chunking compound functional expressions, as well as training/testing of learning dependencies of compound functional expressions, we used manually parsed sentences of Kyoto text corpus [3], that are 38,400 sentences selected from 1995 Mainichi newspaper text. To those data sets, we manually annotate usage labels of the 59 compound functional expressions (details in Table 2).

4.2 Chunking

As we show in Table 3, performance of our SVMs-based chunkers as well as several baselines including existing Japanese text processing tools is evaluated in terms of precision/recall/$F_{\beta=1}$ of identifying *functional* chunks. Performance is evaluated also in terms of accuracy of classifying detected candidate expressions into *functional/content* chunks. Among those baselines, "majority (= *functional*)" always assigns *functional* usage to the detected candidate expressions. Our SVMs-based chunker significantly outperforms those baselines. Some of the errors should be recovered by simply introducing semantic categories of nouns as features.

4.3 Analyzing Dependency Relations

Finally, we evaluate the accuracies of judging dependency relations of compound functional expressions by the variant of CaboCha trained with Kyoto text cor-

pus annotated with usage labels of compound functional expressions. This performance is measured through 10-fold cross validation with the modified version of Kyoto text corpus. Accuracies of identifying modifier(s)/modifiee of compound functional expressions are measured as in Table 4. Here, "CaboCha (w/o FE)" denotes a baseline variant of CaboCha, with all the compound functional expressions removed from its outputs, while "CaoboCha (public)" denotes the publicly available version of CaboCha, which have some portion of the compound functional expressions included in its outputs. For the modifier accuracy, the difference of "chunker + CaboCha (proposed)" and "CaboCha (w/o FE)" is statistically significant at a level of 0.05. Detailed discussions on the evaluation results are in [8].

5 Concluding Remarks

This paper proposed an approach of processing Japanese compound functional expressions by identifying them and analyzing their dependency relations through a machine learning technique. The proposed framework has advantages over an approach based on manually created rules such as the one in [6], in that it requires considerable human effort to manually create and maintain those rules. Future works include the issue of implementing chunkers and dependency analyzers which incorporate much larger number of compound functional expressions with training corpus of limited size.

References

1. T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *Proc. 2nd NAACL*, pages 192–199, 2001.
2. T. Kudo and Y. Matsumoto. Japanese dependency analyisis using cascaded chunking. In *Proc. 6th CoNLL*, pages 63–69, 2002.
3. S. Kurohashi and M. Nagao. Building a Japanese parsed corpus while improving the parsing system. In *Proc. 1st LREC*, pages 719–724, 1998.
4. National Language Research Institute. *Gendaigo Hukugouji Youreishu*. 2001. (in Japanese).
5. I. Sag, T. Baldwin, F. Bond, A. Copestake, and D. Flickinger. Multiword expressions: A pain in the neck for NLP. In *Proc. 3rd CICLING*, pages 1–15, 2002.
6. K. Shudo, T. Tanabe, M. Takahashi, and K. Yoshimura. MWEs as non-propositional content indicators. In *Proc. 2nd ACL Workshop on Multiword Expressions: Integrating Processing*, pages 32–39, 2004.
7. M. Tsuchiya, T. Shime, T. Takagi, T. Utsuro, K. Uchimoto, S. Matsuyoshi, S. Sato, and S. Nakagawa. Chunking Japanese compound functional expressions by machine learning. In *Proc. Workshop on Multi-Word-Expressions in a Multilingual Context*, pages 25–32, 2006.
8. T. Utsuro, T. Shime, M. Tsuchiya, S. Matsuyoshi, and S. Sato. Learning dependency relations of Japanese compound functional expressions. In *Proc. Workshop on A Broader Perspective on Multiword Expressions (ACL-2007 Workshop)*, 2007.
9. V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.