Learning Dependency Relations of Japanese Compound Functional Expressions

Takehito Utsuro† and **Takao Shime**‡ and **Masatoshi Tsuchiya**††
Suguru Matsuyoshi†‡ and **Satoshi Sato**†‡
†Graduate School of Systems and Information Engineering, University of Tsukuba,
1-1-1, Tennodai, Tsukuba, 305-8573, JAPAN
‡NEC Corporation
††Computer Center, Toyohashi University of Technology,
Tenpaku-cho, Toyohashi, 441–8580, JAPAN
†‡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. First, we formalize the task of identifying Japanese compound functional expressions in a text as a machine learning based chunking problem. Next, against the results of identifying compound functional expressions, we apply the method of dependency analysis based on the cascaded chunking model. The results of experimental evaluation show that, the dependency analysis model achieves improvements when applied after identifying compound functional expressions, compared with the case where it is applied without identifying compound functional expressions.

1 Introduction

In addition to single functional words, the Japanese language has many more compound functional expressions which consist of more than one word including both content words and functional words. They are very important for recognizing syntactic structures of Japanese sentences and for understanding their semantic content. Recognition and understanding of them are also very important for various kinds of NLP applications such as dialogue systems, machine translation, and question answering. However, recognition and semantic interpretation of compound functional expressions are especially difficult because it often happens that one compound expression may have both a literal (i.e. compositional) *content word* usage and a non-literal (i.e. non-compositional) *functional* usage.

For example, Table 1 shows two example sentences of a compound expression "t (ni) mMo (tsuite)", which consists of a post-positional particle "t (ni)", and a conjugated form "mMo (tsuite)" of a verb "mX (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 "t (ni)" and the verb "mMo (tsuite)", and has a content word meaning "*follow*". Therefore, when considering machine translation of these Japanese sentences into English, it is necessary to judge precisely the usage of the compound expression "t (ni) mMo (tsuite)", as shown in the English translation of the two sentences in Table 1.

There exist widely-used Japanese text processing tools, i.e. combinations of a morphological analysis tool and a subsequent parsing tool, such as JUMAN[1]+ KNP[2] and ChaSen[3]+ CaboCha[4]. However, they process those compound expressions only partially, in that their morphological analysis dictionaries list only a 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 ambigui-

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

Proceedings of the Workshop on A Broader Perspective on Multiword Expressions, pages 65–72,
Prague, June 2007. ©2007 Association for Computational Linguistics

(A)	(watashi) x (ha) t (kare) t (ni) mMo (tsuite) ê h (hanashita) (*I*) (*TOP*) (*he*) (*about*) (*talked*) (I talked about him.)
(B)	² (watashi) x (ha) t (kare) t (ni) mMo (tsuite) ø h (hashitta) (*I*) (*TOP*) (*he*) (*ACC*) (*follow*) (*ran*) (I ran following him.)

Table 1: Translation Selection of a Japanese Compound Expression "t (ni) mMo (tsuite)"

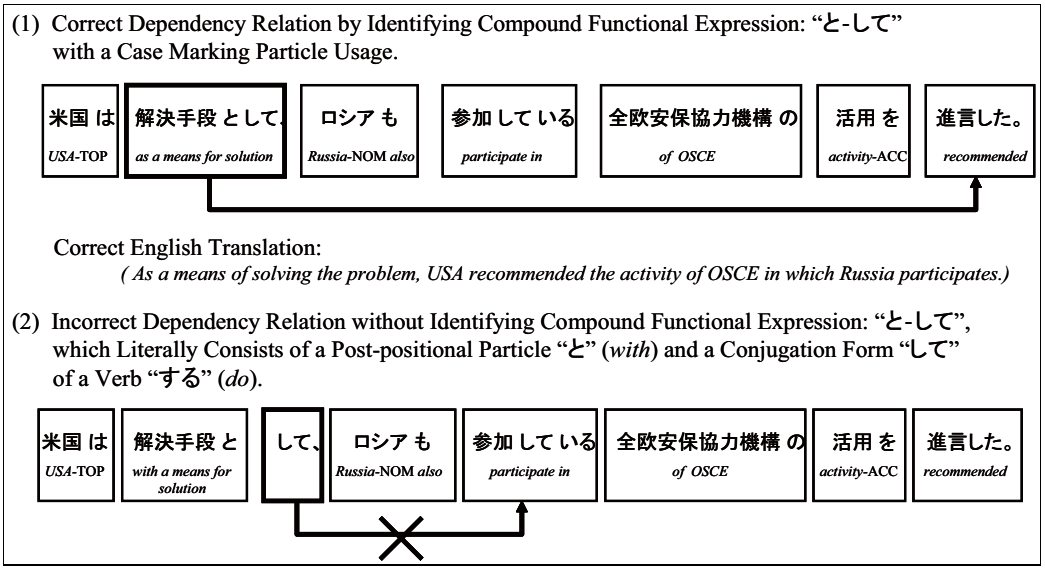


Figure 1: Example of Improving Dependency Analysis of Compound Functional Expressions by Identifying them before Dependency Analysis

ties of their usages, such as those in Table 1. This is mainly because the framework of these 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 we introduce in the next section, as a first step towards studying computational processing of compound functional expressions, we start with 125 major functional expressions which have non-compositional usages, as well as their variants (337 expressions in total). Out of those 337 expressions, 111 have both a *content word* usage and a *functional* usage. However, the combination of JUMAN+KNP is capable of distinguishing the two usages only for 43 of the 111 expressions, and the combination 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, (1) of Figure 1 gives an example of identifying a correct modifiee of the second *bunsetsu* segment [5] "r>/^q`o (*as a means for solution*)" including a Japanese compound functional expression "q`o (*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 third bunsetsu "`o ", which actually happens if we do not identify the compound functional expression "q`o (*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 il-

[5] A Japanese *bunsetsu* segment is a phrasal unit which consits of at least one content word and zero or more functional words.

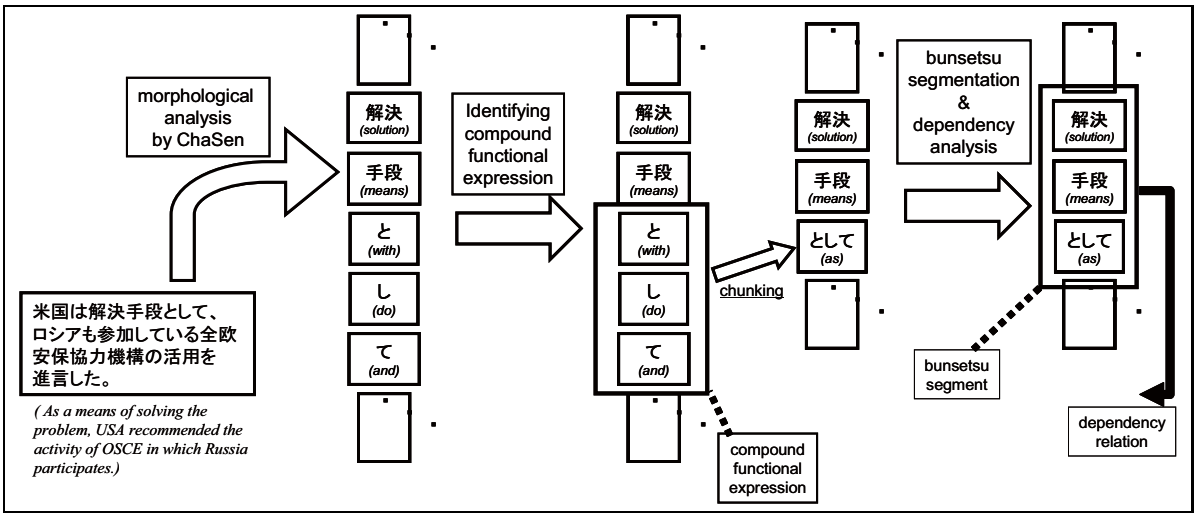


Figure 2: Overall Flow of Processing Compound Functional Expressions in a Japanese Sentence

lustrated 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 their dependency relations. We formalize the task of identifying Japanese compound functional expressions in a text as a machine learning based chunking problem (Tsuchiya et al., 2006). We employ the technique of Support Vector Machines (SVMs) (Vapnik, 1998) as the machine learning technique, which has been successfully applied to various natural language processing tasks including chunking tasks such as phrase chunking 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 (Kudo and Matsumoto, 2002), which is simple and efficient because it parses a sentence deterministically only deciding whether the current bunsetsu segment modifies the one on its immediate right hand side. As we showed in Figure 1, identifying compound functional expressions before analyzing dependencies in a sentence does actually help deciding dependency relations of compound functional expressions.

In the experimental evaluation, we focus on 59 expressions having 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. We first show that the proposed method of chunking compound functional expressions significantly outperforms existing Japanese text processing tools. Next, we further show that the dependency analysis model of (Kudo and Matsumoto, 2002) applied to the results of identifying compound functional expressions significantly outperforms the one applied to the results without identifying compound functional expressions.

2 Japanese Compound Functional Expressions

There exist several collections which list Japanese functional expressions and examine their usages. For example, (Morita and Matsuki, 1989) examine 450 functional expressions and (Group Jamashii, 1998) also lists 965 expressions and their example sentences. Compared with those two collections, *Gendaigo Hukugouji Youreishu* (National Language Research Institute, 2001) (henceforth, denoted as *GHY*) concentrates on 125 major functional expressions which have non-compositional usages, as well as their variants[6], and collects example sentences of those expressions. As we mentioned in the previous section, as a first step towards developing a tool for identifying Japanese compound functional expressions, we start with those 125 major functional expressions and their variants (337 expressions in to-

[6]For each of those 125 major expressions, the differences between it and its variants are summarized as below: i) insertion/deletion/alternation of certain particles, ii) alternation of synonymous words, iii) normal/honorific/conversational forms, iv) base/adnominal/negative forms.

(a) Classification of Compound Functional Expressions based on Grammatical Function

Grammatical Function Type		# of major expressions	# of variants	Example
post-positional particle type	conjunctive particle	36	67	Xdt (kuse-ni)
	case-marking particle	45	121	q`o (to-shite)
	adnominal particle	2	3	qMO (to-iu)
auxiliary verb type		42	146	oMM (te-ii)
total		125	337	—

(b) Examples of Classifying Functional/Content Usages

	Expression	Example sentence (English translation)	Usage
(1)	Xdt (kuse-ni)	txÚ• Xdt zS•tx⁄›\hiZi{ (To my brother, (someone) gave money, *while* (he/she) did nothing to me but just sent a letter.)	functional (Xdt (kuse-ni) = *while*)
(2)	Xdt (kuse-ni)	twfw Xdt ˆœs»Mh{ (They all were surprised *by* his *habit*.)	content (∼ →→→ (kuse-ni) = *by one's habit*
(3)	q`o (to-shite)	txfwðJw•óH q`o Œ'•oM"{ (He is known *as* an expert of the problem.)	functional (∼ →→→ (to-shite) = *as* ∼)
(4)	q`o (to-shite)	\•UY`MTrOTxlV" q`o <^M{ (Please *make* it clear whether this is true or not.)	content (∼ → ∼ →→→ (to-shite) = *make* ∼ ∼
(5)	qMO (to-iu)	tx\VoM" qMO Œ'dÞ Mh{ (I heard *that* he is alive.)	functional (∼ →→→ (to-iu) = *that* ∼)
(6)	qMO (to-iu)	®|tRo<^M¯ qMO ‰M"{ (Somebody *says* "Please visit us.".)	content (∼ →→→ (to-iu) = *say (that)* ∼)
(7)	oMM (te-ii)	\w^æU4lh'sˏ oMM { (You *may* have a break after we finish this discussion.)	functional (∼ →→→ (te-ii) = *may* ∼)
(8)	oMM (te-ii)	\wTyœxGVX oMM { (This bag is *nice because* it is big.)	content (∼ →→→ (te-ii) = *nice because* ∼)

Table 2: Classification and Example Usages of Compound Functional Expressions

tal). In this paper, following (Sag et al., 2002), we regard each variant as a fixed expression, rather than a semi-fixed expression or a syntactically-flexible expression [7]. Then, we focus on evaluating the effectiveness of straightforwardly applying a standard chunking technique to the task of identifying Japanese compound functional expressions.

As in Table 2 (a), according to their grammatical functions, those 337 expressions in total are roughly classified into *post-positional particle* type, and *auxiliary verb* type. Functional expressions of post-positional particle type are further classified into three subtypes: i) conjunctive particle types, which are used for constructing subordinate clauses, ii) case-marking particle types, iii) adnominal particle types, which are used for constructing adnominal clauses. Furthermore, for examples of compound functional expressions listed in Table 2 (a), Table 2 (b) gives their example sentences as well as the description of their usages.

3 Identifying Compound Functional Expressions by Chunking with SVMs

This section describes summaries of formalizing the chunking task using SVMs (Tsuchiya et al., 2006). In this paper, we use an SVMs-based chunking tool YamCha[8] (Kudo and Matsumoto, 2001). 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

[7] Compound functional expressions of auxiliary verb types can be regarded as syntactically-flexible expressions.

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

sentence is represented as a sequence of morphemes, where a morpheme is regarded as a token.

3.1 Chunk Representation

For representing proper chunks, we employ IOB2 representation, which has been studied well in various chunking tasks of natural language processing. This method uses the following set of three labels for representing proper chunks.

- **I** Current token is a middle or the end of a chunk consisting of more than one token.
- **O** Current token is outside of any chunk.
- **B** Current token is the beginning of a chunk.

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. In total, we have the following five labels for representing those chunks: **B-functional, I-functional, B-content, I-content**, and **O**. Finally, as for extending SVMs to multi-class classifiers, we experimentally compare the *pairwise* method and the *one vs. rest* method, where the *pairwise* method slightly outperformed the *one vs. rest* method. Throughout the paper, we show results with the *pairwise* method.

3.2 Features

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 (Tsuchiya et al., 2006), which mainly consists of morphemes as well as information on the candidate compound functional expression at i-th position.

4 Learning Dependency Relations of Japanese Compound Functional Expressions

4.1 Japanese Dependency Analysis using Cascaded Chunking

4.1.1 Cascaded Chunking Model

First of all, we define a Japanese sentence as a sequence of bunsetsu segments $B = \langle b_1, b_2, \ldots, b_m \rangle$ and its syntactic structure as a sequence of dependency patterns $D = \langle Dep(1), Dep(2), \ldots, Dep(m-1) \rangle$, where $Dep(i) = j$ means that the bunsetsu segment b_i depends on (modifies) bunsetsu segment b_j. In this framework, we assume that the dependency sequence D 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.

Unlike probabilistic dependency analysis models of Japanese, the cascaded chunking model of Kudo and Matsumoto (2002) does not require the probabilities of dependencies and parses a sentence deterministically. Since Japanese is a head-final language, and the chunking can be regarded as the creation of a dependency between two bunsetsu segments, this model simplifies the process of Japanese dependency analysis as follows: [9]

1. Put an **O** tag on all bunsetsu segments. The **O** tag indicates that the dependency relation of the current segment is undecided.
2. For each bunsetsu segment with an **O** tag, decide whether it modifies the bunsetsu segment on its immediate right hand side. If so, the **O** tag is replaced with a **D** tag.
3. Delete all bunsetsu segments with a **D** tag that immediately follows a bunsetsu segment with an **O** tag.

[9] The **O** and **D** tags used in this section have no relation to those chunk reppresentation tags introduced in section 3.1.

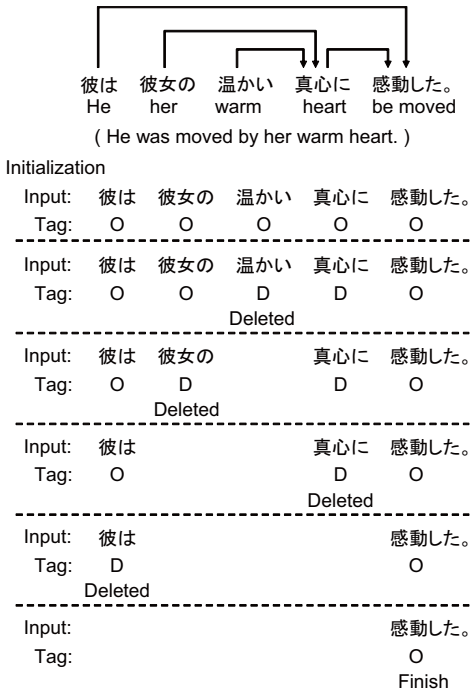


Figure 3: Example of the Parsing Process with Cascaded Chunking Model

4. Terminate the algorithm if a single bunsetsu segment remains, otherwise return to the step 2 and repeat.

Figure 3 shows an example of the parsing process with the cascaded chunking model.

4.1.2 Features

As a Japanese dependency analyzer based on the cascaded chunking model, we use the publicly available version of CaboCha (Kudo and Matsumoto, 2002), which is trained with the manually parsed sentences of Kyoto text corpus (Kurohashi and Nagao, 1998), that are 38,400 sentences selected from the 1995 Mainichi newspaper text.

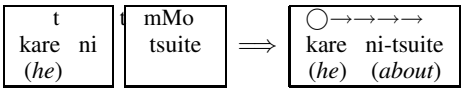
The standard feature set used by CaboCha consists of **static features** and **dynamic features**. Static features are those solely defined once the pair of modifier/modifiee bunsetsu segments is specified. For the pair of modifier/modifiee bunsetsu segments, the following are used as static features: 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. 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 as static features. On the other hand, dynamic features are created during the parsing process, so that, when a certain dependency relation is determined, it can have some influence on other dependency relations. Dynamic features include bunsetsu segments modifying the current candidate modifiee (see Kudo and Matsumoto (2002) for the details).

4.2 Coping with Compound Functional Expressions

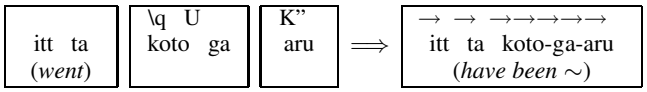
As we show in Figure 2, a compound functional expression is identified as a sequence of several morphemes and then chunked into one morpheme. The result of this identification process is then transformed into the sequence of bunsetsu segments. Finally, to this modified sequence of bunsetsu segments, 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 “t (ni) mMo (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.



(B) As we show below, a compound functional expression “\q (koto) U (ga) K” (aru)” overlaps the boundary of two bunsetsu segments, though the two bunsetsu segments concatenating into one bunsetsu segment does include no content words. In such a case, its immediate left bunsetsu segment (“at (itt) h (ta)“ in the example below), which corresponds to the content word part of “\q (koto) U (ga) K” (aru)”, has to be concatenated into the bunsetsu segment “\q (koto) U (ga) K” (aru)”.



Next, to the compound functional expression, we assign one of the four grammatical function types listed in Table 2 as its POS tag. For example, the compound functional expression “t (ni) mMo (tsuite)” in (A) above is assigned the grammatical function type “case-marking particle type”, while “ \q (koto) U (ga) K” (aru)” in (B) is assigned “auxiliary verb type”.

These modifications cause differences in the final feature representations. 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 “r>/q`o ” which has the compound functional expression “q`o ” in its functional word part. On the other hand, in (2), the modifier bunsetsu segment is “`o ”, which corresponds to the literal verb usage of a part of the compound functional expression “q`o ”. In the final feature representations below, this causes the following differences in head words and functional words / POS of the modifier bunsetsu segments:

	(1) of Figure 1	(2) of Figure 1
head word	/ (*means*)	b” (*do*)
functional word POS	q`o (*as*) subsequent to nominal / modifying predicate	o (*and*) conjunctive particle

5 Experimental Evaluation

5.1 Training/Test Data Sets

For the training of chunking compound functional expressions, we collected 2,429 example sentences from the 1995 Mainichi newspaper text corpus. For each of the 59 compound functional expressions for evaluation mentioned in section 1, at least 50 examples are included in this training set. 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 (Kurohashi and Nagao, 1998), that are 38,400 sentences selected from the 1995 Mainichi newspaper text (the 2,429 sentences above are selected so that they are exclusive of the 37,400 sentences of Kyoto text corpus.). To those data sets, we manually annotate usage labels of the 59 compound functional expressions (details in Table 3).

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

Table 3: Statistics of Data Sets

	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: Evaluation Results of Chunking (%)

5.2 Chunking

As we show in Table 4, 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 all the 5,744 *functional* chunks included in the test data (Kyoto text corpus in Table 3). 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. Performance of our SVMs-based chunkers is measured through 10-fold cross validation. Our SVMs-based chunker significantly outperforms those baselines both in $F_{\beta=1}$ and classification accuracy. As we mentioned in section 1, existing Japanese text processing tools process compound functional expressions only partially, which causes damage in recall in Table 4.

5.3 Analyzing Dependency Relations

We evaluate the accuracies of judging dependency relations of compound functional expressions by the variant of CaboCha trained with Kyoto text corpus annotated with usage labels of compound functional expressions. This performance is measured through 10-fold cross validation with the modified version of the Kyoto text corpus. In the evaluation phase, according to the flow of Figure 2, first we apply the chunker of compound functional expressions trained with all the 2,429 sentences in Table 3 and obtain the results of chunked compound functional expressions with about 90% correct rate. Then, bunsetsu segmentation and dependency analysis are per-

		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

Table 5: Accuracies of Identifying Modifier(s)/Modifiee (%)

formed by our variant of CaboCha, where accuracies of identifying modifier(s)/modifiee of compound functional expressions are measured as in Table 5 ("chunker + CaboCha (proposed)" denotes that inputs to CaboCha (proposed) are with 90% correct rate, while "reference + CaboCha (proposed)" denotes that they are with 100% correct rate). Here, "CaboCha (w/o FE)" denotes a baseline variant of CaboCha, with all the compound functional expressions removed from its inputs (which are outputs from ChaSen), while "CaoboCha (public)" denotes the publicly available version of CaboCha, which have some portion of the compound functional expressions included in its inputs.

For the modifier accuracy, the difference of "chunker + CaboCha (proposed)" and "CaboCha (w/o FE)" is statistically significant at a level of 0.05. Identifying compound functional expressions typically contributes to improvements when the literal constituents of a compound functional expression include a verb. In such a case, for bunsetsu segments which usually modifies a verb, an incorrect modifee candidate is removed, which results in improvements in the modifier accuracy. The difference between 'CaoboCha (public)" and "chunker + CaboCha (proposed)" is slight because the publicly available version of CaboCha seems to include compound functional expressions which are damaged in identifying their modifiers with "CaboCha (w/o FE)". For the modifiee accuracy, the difference of "chunker + CaboCha (proposed)" and "CaboCha (w/o FE)" is zero. Here, more than 100 instances of improvements like the one in Figure 1 are observed, while almost the same number of additional failures are also observed mainly because of the sparseness problem. Furthermore, in the case of the modifiee accuracy, it is somehow difficult to expect improvement because identifying modifiees of *functional/content* bunsetsu segments mostly depends on features other than *functional/content* distinction.

6 Concluding Remarks

We proposed an approach of processing Japanese compound functional expressions by identifying them and analyzing their dependency relations through a machine learning technique. This approach is novel in that it has never been applied to any language so far. Experimental evaluation showed 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. The proposed framework has advantages over an approach based on manually created rules such as the one in (Shudo et al., 2004), in that it requires human cost to create manually and maintain those rules. Related works include Nivre and Nilsson (2004), which reports improvement of Swedish parsing when multi word units are manually annotated.

References

Group Jamashii, editor. 1998. *Nihongo Bunkei Jiten*. Kuroshio Publisher. (in Japanese).

T. Kudo and Y. Matsumoto. 2001. Chunking with support vector machines. In *Proc. 2nd NAACL*, pages 192–199.

T. Kudo and Y. Matsumoto. 2002. Japanese dependency analyisis using cascaded chunking. In *Proc. 6th CoNLL*, pages 63–69.

S. Kurohashi and M. Nagao. 1998. Building a Japanese parsed corpus while improving the parsing system. In *Proc. 1st LREC*, pages 719–724.

Y. Morita and M. Matsuki. 1989. *Nihongo Hyougen Bunkei*, volume 5 of *NAFL Sensho*. ALC. (in Japanese).

National Language Research Institute. 2001. *Gendaigo Hukugouji Youreishu*. (in Japanese).

J. Nivre and J. Nilsson. 2004. Multiword units in syntactic parsing. In *Proc. LREC Workshop, Methodologies and Evaluation of Multiword Units in Real-World Applications*, pages 39–46.

I. Sag, T. Baldwin, F. Bond, A. Copestake, and D. Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proc. 3rd CICLING*, pages 1–15.

K. Shudo, T. Tanabe, M. Takahashi, and K. Yoshimura. 2004. MWEs as non-propositional content indicators. In *Proc. 2nd ACL Workshop on Multiword Expressions: Integrating Processing*, pages 32–39.

M. Tsuchiya, T. Shime, T. Takagi, T. Utsuro, K. Uchimoto, S. Matsuyoshi, S. Sato, and S. Nakagawa. 2006. Chunking Japanese compound functional expressions by machine learning. In *Proc. Workshop on Multi-Word-Expressions in a Multilingual Context*, pages 25–32.

V. N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.