Chunking Japanese Compound Functional Expressions by Machine Learning

Masatoshi Tsuchiya† and Takao Shime‡ and Toshihiro Takagi†
Takehito Utsuro†† and Kiyotaka Uchimoto†† and Suguru Matsuyoshi†
Satoshi Sato‡‡ and Seiichi Nakagawa‡‡

†Computer Center / ‡‡Department of Information and Computer Sciences, Toyohashi University of Technology, Tenpaku-cho, Toyohashi, 441–8580, JAPAN
‡Graduate School of Informatics, Kyoto University, Sakyo-ku, Kyoto, 606–8501, JAPAN
††Graduate School of Systems and Information Engineering, University of Tsukuba, 1-1-1, Tennodai, Tsukuba, 305-8573, JAPAN
‡‡National Institute of Information and Communications Technology, 3–5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619–0289 JAPAN

Abstract
The Japanese language has various types of compound functional expressions, which are very important for recognizing the syntactic structures of Japanese sentences and for understanding their semantic contents. In this paper, we formalize the task of identifying Japanese compound functional expressions in a text as a chunking problem. We apply a machine learning technique to this task, where we employ that of Support Vector Machines (SVMs). We show that the proposed method significantly outperforms existing Japanese text processing tools.

1 Introduction
As in the case of other languages, the Japanese language has various types of functional words such as post-positional particles and auxiliary verbs. In addition to those functional words, the Japanese language has much more compound functional expressions which consist of more than one words including both content words and functional words. Those single functional words as well as compound functional expressions are very important for recognizing the syntactic structures of Japanese sentences and for understanding their semantic contents. 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 (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 “に (ni) ついで (tsuite)”, which consists of a post-positional particle “に (ni)”, and a conjugated form “ついで (tsuite)” of a verb “つぐ (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 “に (ni)” and the verb “ついで (tsuite)”, and has a content word meaning “follow”. Therefore, when considering machine translation of those Japanese sentences into English, it is necessary to precisely judge the usage of the compound expression “に (ni) ついで (tsuite)”, as shown in the English translation of the two sentences in Table 1.

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+ KNP2 and ChaSen3+ CaboCha4. 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

1http://www.kc.t.u-tokyo.ac.jp/nl-resource/juman-e.html
2http://www.kc.t.u-tokyo.ac.jp/nl-resource/knp-e.html
3http://chasen.naist.jp/hiki/ChaSen/
4http://chasen.org/~taku/software/cabocha/
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.

Considering such a situation, it is necessary to develop a tool which properly recognizes and semantically interprets Japanese compound functional expressions. In this paper, we apply a machine learning technique to the task of identifying Japanese compound functional expressions in a text. We formalize this identification task as a chunking problem. 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 (Kudo and Matsumoto, 2001) and named entity chunking (Mayfield et al., 2003).

In the preliminary experimental evaluation, we focus on 52 expressions that 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. We show that the proposed method significantly outperforms existing Japanese text processing tools as well as another tool based on hand-crafted rules. We further show that, in the proposed SVMs based framework, it is sufficient to collect and manually annotate about 50 training examples per expression.

2 Japanese Compound Functional Expressions and their Example Database

2.1 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\(^5\) (337 expressions in total), and collects example sentences of those expressions. As a first step of developing a tool for identifying Japanese compound functional expressions, we start with those 125 major functional expressions and their variants. 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 (Sag et al., 2002). Then, we focus on evaluating the effectiveness of straightforwardly applying a stan-

\(^5\)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.
<table>
<thead>
<tr>
<th>Expression</th>
<th>Example sentence (English translation)</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) とならない (to-naru-to)</td>
<td>しっかりこの病気に効果がないにすると、事は重大だ。 (The situation is serious if it is not effective against this disease.)</td>
<td>functional</td>
</tr>
<tr>
<td>(2) とならない (to-naru-to)</td>
<td>彼が社長になるための条件の一つとなると考えられている。 (They think that it will become a requirement for him to be the president.)</td>
<td>content</td>
</tr>
<tr>
<td>(3) にかけては (ni-kakete-ha)</td>
<td>お金を儲けることに関しては、素晴らしい才能をもっている。 (He has a great talent for earning money.)</td>
<td>functional</td>
</tr>
<tr>
<td>(4) にかけては (ni-kakete-ha)</td>
<td>あまり忙しくてはいない。 (I do not worry about it.)</td>
<td>content</td>
</tr>
<tr>
<td>(5) という (to-iu)</td>
<td>彼は生きているという知らせを聞いた。 (I heard that he is alive.)</td>
<td>functional</td>
</tr>
<tr>
<td>(6) という (to-iu)</td>
<td>「遊びに来て下さい」という人もいる。 (Somebody says “Please visit us.”)</td>
<td>content</td>
</tr>
<tr>
<td>(7) ていい (te-ii)</td>
<td>この議論が終わったら休憩してよい。 (You may have a break after we finish this discussion.)</td>
<td>functional</td>
</tr>
<tr>
<td>(8) ていい (te-ii)</td>
<td>このかばんは大きい。 (This bag is nice because it is big.)</td>
<td>content</td>
</tr>
</tbody>
</table>

As in Table 2, 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) those subsequent to a predicate and modifying a predicate, which mainly function as conjunctive particles and are used for constructing subordinate clauses, ii) those subsequent to a nominal, and modifying a predicate, which mainly function as case-marking particles, iii) those subsequent to a nominal, and modifying a nominal, which mainly function as adnominal particles and are used for constructing adnominal clauses. For each of those types, Table 2 also shows the number of major expressions as well as that of their variants listed in GHY, and an example expression. Furthermore, Table 3 gives example sentences of those example expressions as well as the description of their usages.

### 2.2 Issues on Identifying Compound Functional Expressions in a Text

The task of identifying Japanese compound functional expressions roughly consists of detecting candidates of compound functional expressions in a text and of judging the usages of those candidate expressions. The class of Japanese compound functional expressions can be regarded as closed and their number is at most a few thousand.
Table 4: Examples of Detecting more than one Candidate Expression

<table>
<thead>
<tr>
<th>Expression</th>
<th>Example sentence (English translation)</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>(9) とう (to-iu)</td>
<td>それが試合というものの難しさだ。 (That’s why a match is not so easy.)</td>
<td>functional $(NP_1 とう (to-iu))NP_2 = NP_2$ called as $NP_1$</td>
</tr>
<tr>
<td>(10) というもののも (to-iu-mono-no)</td>
<td>勝ったというもののも、スコアは悪い。 (Although he won, the score is bad.)</td>
<td>functional $(∼ というもののも$ (to-iu-mono-no) $∼ = although ∼)$</td>
</tr>
</tbody>
</table>

Therefore, it is easy to enumerate all the compound functional expressions and their morpheme sequences. Then, in the process of detecting candidates of compound functional expressions in a text, the text are matched against the morpheme sequences of the compound functional expressions considered.

Here, most of the 125 major functional expressions we consider in this paper are compound expressions which consist of one or more content words as well as functional words. As we introduced with the examples of Table 1, it is often the case that they have both a compositional content word usage as well as a non-compositional functional usage. For example, in Table 3, the expression “なると (to-naru-to)” in the sentence (2) has the meaning “that (something) becomes ∼”, which corresponds to a literal concatenation of the usages of the constituents: the post-positional particle “と”, the verb “なる”, and the post-positional particle “と”, and can be regarded as a content word usage. On the other hand, in the case of the sentence (1), the expression “なると (to-naru-to)” has a non-compositional functional meaning “if”. Based on this discussion, we classify the usages of those expressions into two classes: functional and content. Here, functional usages include both non-compositional and compositional functional usages, although most of the functional usages of those 125 major expressions can be regarded as non-compositional. On the other hand, content usages include compositional content word usages only.

More practically, in the process of detecting candidates of compound functional expressions in a text, it can happen that more than one candidate expression is detected. For example, in Table 4, both of the candidate compound functional expressions “とう (to-iu)” and “というもののも (to-iu-mono-no)” are detected in the sentence (9). This is because the sequence of the two morphemes “と (to)” and “いう (iu)” constituting the candidate expression “とう (to-iu)” is a subsequence of the four morphemes constituting the candidate expression “というもののも (to-iu-mono-no)” as below:

<table>
<thead>
<tr>
<th>Morpheme sequence</th>
<th>Candidate expression</th>
<th>Candidate expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>と (to) いう (iu)</td>
<td>とう (to-iu)</td>
<td>というもののも (to-iu-mono-no)</td>
</tr>
<tr>
<td>もの (mono) の (no)</td>
<td>と (to) いう (iu)</td>
<td>とも (to-iu) もの (mono) の (no)</td>
</tr>
</tbody>
</table>

This is also the case with the sentence (10).

Here, however, as indicated in Table 4, the sentence (9) is an example of the functional usage of the compound functional expression “とう (to-iu)”, where the sequence of the two morphemes “と (to)” and “いう (iu)” should be identified and chunked into a compound functional expression. On the other hand, the sentence (10) is an example of the functional usage of the compound functional expression “というもののも (to-iu-mono-no)”, where the sequence of the four morphemes “と (to)”, “いう (iu)”, “もの (mono)”, and “の (no)” should be identified and chunked into a compound functional expression. Actually, in the result of our preliminary corpus study, at least in about 20% of the occurrences of Japanese compound functional expressions, more than one candidate expression can be detected. This result indicates that it is necessary to consider more than one candidate expression in the task of identifying a Japanese compound functional expression, and also in the task of classifying the functional/content usage of a candidate expression. Thus, in this paper, based on this observation, we formalize the task of identifying Japanese compound functional expressions as a chunking problem, rather than a classification problem.
Table 5: Number of Sentences collected from 1995 Mainichi Newspaper Texts (for 337 Expressions)

<table>
<thead>
<tr>
<th># of sentences</th>
<th># of expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 ≤ #</td>
<td>187 (55%)</td>
</tr>
<tr>
<td>0 &lt; # &lt; 50</td>
<td>117 (35%)</td>
</tr>
<tr>
<td># = 0</td>
<td>33 (10%)</td>
</tr>
</tbody>
</table>

2.3 Developing an Example Database

We developed an example database of Japanese compound functional expressions, which is used for training/testing a chunker of Japanese compound functional expressions (Tsuchiya et al., 2005). The corpus from which we collect example sentences is 1995 Mainichi newspaper text corpus (1,294,794 sentences, 47,355,330 bytes). For each of the 337 expressions, 50 sentences are collected and chunk labels are annotated according to the following procedure.

1. The expression is morphologically analyzed by ChaSen, and its morpheme sequence⁴ is obtained.
2. The corpus is morphologically analyzed by ChaSen, and 50 sentences which include the morpheme sequence of the expression are collected.
3. For each sentence, every occurrence of the 337 expressions is annotated with one of the usages functional/content by an annotator⁷.

Table 5 classifies the 337 expressions according to the number of sentences collected from the 1995 Mainichi newspaper text corpus. For more than half of the 337 expressions, more than 50 sentences are collected, although about 10% of the 377 expressions do not appear in the whole corpus. Out of those 187 expressions with more than 50 sentences, 52 are those with balanced distribution of the functional/content usages in the newspaper text corpus. Those 52 expressions can be regarded as among the most difficult ones in the task of identifying and classifying functional/content

---

⁴For those expressions whose constituent has conjugation and the conjugated form also has the same usage as the expression with the original form, the morpheme sequence is expanded so that the expanded morpheme sequences include those with conjugated forms.

⁷For the most frequent 184 expressions, on the average, the agreement rate between two human annotators is 0.95 and the Kappa value is 0.73, which means allowing tentative conclusions to be drawn (Carletta, 1996; Ng et al., 1999). For 65% of the 184 expressions, the Kappa value is above 0.8, which means good reliability.

3 Chunking Japanese Compound Functional Expressions with SVMs

3.1 Support Vector Machines

The principle idea of SVMs is to find a separate hyperplane that maximizes the margin between two classes (Vapnik, 1998). If the classes are not separated by a hyperplane in the original input space, the samples are transformed in a higher dimensional features space.

Giving x is the context (a set of features) of an input example; xᵢ and yᵢ (i = 1,...,l, xᵢ ∈ Rⁿ, yᵢ ∈ {1,−1}) indicate the context of the training data and its category, respectively; The decision function f in SVM framework is defined as:

\[ f(x) = \text{sgn} \left( \sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b \right) \]  

where K is a kernel function, b ∈ R is a threshold, and αᵢ are weights. Besides, the weights αᵢ satisfy the following constraints:

\[ 0 \leq \alpha_i \leq C \quad (i = 1, \ldots, l) \]  
\[ \sum_{i=1}^{l} \alpha_i y_i = 0 \]  

where C is a misclassification cost. The xᵢ with non-zero αᵢ are called support vectors. To train an SVM is to find the αᵢ and the b by solving the optimization problem; maximizing the following under the constraints of (2) and (3):

\[ L(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \]  

The kernel function K is used to transform the samples in a higher dimensional features space. Among many kinds of kernel functions available, we focus on the d-th polynomial kernel:

\[ K(x, y) = (x \cdot y + 1)^d \]
Through experimental evaluation on chunking Japanese compound functional expressions, we compared polynomial kernels with \( d = 1, 2, \) and 3. Kernels with \( d = 2 \) and 3 perform best, while the kernel with \( d = 3 \) requires much more computational cost than that with \( d = 2 \). Thus, throughout the paper, we show results with the quadratic kernel \((d = 2)\).

### 3.2 Chunking with SVMs

This section describes details of formalizing the chunking task using SVMs. 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 sentence is represented as a sequence of morphemes, where a morpheme is regarded as a token.

#### 3.2.1 Chunk Representation

For representing proper chunks, we employ IOB2 representation, one of those which have been studied well in various chunking tasks of natural language processing (Tjong Kim Sang, 1999; Kudo and Matsumoto, 2001). 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.

As described in section 2.2, 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. Table 6 gives examples of those chunk labels representing chunks.

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.

\(^8\)http://chasen.org/~taku/software/yamcha/

#### 3.2.2 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:

- **MORPH** \( m_i \): Morpheme appearing at \( i \)-th position.
- **Parsing Direction** \( D \): Parsing direction.
- **Feature set** \( F \): Feature set at a position, chunk label \( c_i \).

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 \( c_{i-1} \).

For representing proper chunks, we employ the following set of three labels for representing proper chunks:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I</strong></td>
<td><strong>O</strong></td>
</tr>
<tr>
<td><strong>B</strong></td>
<td><strong>O</strong></td>
</tr>
</tbody>
</table>

As described in section 2.2, 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. Table 6 gives examples of those chunk labels representing chunks.

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.

8http://chasen.org/~taku/software/yamcha/
as well as the chunk candidate features at immediate left/right contexts of \( E \).

\[
CF(i) = \langle \text{length of } E, \text{position of } m_i \text{ in } E \rangle \\
OF(i) = \langle MF(m_{j-2}), CF(j-2), \\
MF(m_{j-1}), CF(j-1), \\
MF(m_{k+1}), CF(k+1),
MF(m_{k+2}), CF(k+2) \rangle
\]

Table 6 gives examples of chunk candidate features and chunk context features.

It can happen that the morpheme at the current position \( i \) constitutes more than one candidate compound functional expression. For example, in the example below, the morpheme sequences \( m_{i-1} m_i m_{i+1} \) and \( m_{i-1} m_i m_{i+2} \) constitute candidate expressions \( E_1 \), \( E_2 \), and \( E_3 \), respectively.

\[
\begin{align*}
\text{Morpheme sequence} & : m_{i-1} m_i m_{i+1} m_{i+2} \\
\text{Candidate } E_1 & : m_{i-1} m_i m_{i+1} \\
\text{Candidate } E_2 & : m_{i-1} m_i m_{i+2} \\
\text{Candidate } E_3 & : \underbrace{m_i m_{i+1} m_{i+2}}_{m_{i-1}}
\end{align*}
\]

In such cases, we prefer the one starting with the leftmost morpheme. If more than one candidate expression starts with the leftmost morpheme, we prefer the longest one. In the example above, we prefer the candidate \( E_1 \) and construct the chunk candidate features and chunk context features considering \( E_1 \) only.

4 Experimental Evaluation

The detail of the data set we use in the experimental evaluation was presented in section 2.3. As we show in Table 7, 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\(_{\beta}\) 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. “Hand-crafted rules” are manually created 145 rules each of which has conditions on morphemes constituting a compound functional expression as well as those at immediate left/right contexts. Performance of our SVMs-based chunkers is measured through 10-fold cross validation.

As shown in Table 7, our SVMs-based chunkers significantly outperform those baselines both in F\(_{\beta}\) and classification accuracy\(^9\). We also evaluate the effectiveness of each feature set, i.e., the morpheme feature, the chunk candidate feature, and the chunk context feature. The results in the table show that the chunker with the chunk candidate feature performs almost best even without the chunk context feature\(^10\).

\(^9\)Recall of existing Japanese text processing tools is low, because those tools can process only 50–60% of the whole 52 compound functional expressions, and for the remaining 40–50% expressions, they fail in identifying all of the occurrences of functional usages.

\(^10\)It is also worthwhile to note that training the SVMs-based chunker with the full set of features requires computational cost three times as much as training without the chunk
Table 7: Evaluation Results (%)

<table>
<thead>
<tr>
<th>Baselines</th>
<th>Prec.</th>
<th>Rec.</th>
<th>$F_{\beta=1}$</th>
<th>Acc. of classifying functional/content chunks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority (= functional)</td>
<td>78.0</td>
<td>100</td>
<td>87.6</td>
<td>78.0</td>
</tr>
<tr>
<td>Juman/KNP</td>
<td>89.2</td>
<td>49.3</td>
<td>63.5</td>
<td>55.8</td>
</tr>
<tr>
<td>ChaSen/CaboCha</td>
<td>89.0</td>
<td>45.6</td>
<td>60.3</td>
<td>53.2</td>
</tr>
<tr>
<td>hand-crafted rules</td>
<td>90.7</td>
<td>81.6</td>
<td>85.9</td>
<td>79.1</td>
</tr>
<tr>
<td>SVM (feature set)</td>
<td>88.0</td>
<td>91.0</td>
<td>89.4</td>
<td>86.5</td>
</tr>
<tr>
<td>morpheme</td>
<td>91.0</td>
<td>93.2</td>
<td><strong>92.1</strong></td>
<td><strong>89.0</strong></td>
</tr>
<tr>
<td>morpheme + chunk-candidate</td>
<td>91.1</td>
<td>93.6</td>
<td><strong>92.3</strong></td>
<td><strong>89.2</strong></td>
</tr>
</tbody>
</table>

Figure 1: Change of $F_{\beta=1}$ with Different Number of Training Instances

For the SVMs-based chunker with the chunk candidate feature with/without the chunk context feature, Figure 1 plots the change of $F_{\beta=1}$ when training with different number of labeled chunks as training instances. With this result, the increase in $F_{\beta=1}$ seems to stop with the maximum number of training instances, which supports the claim that it is sufficient to collect and manually annotate about 50 training examples per expression.

5 Concluding Remarks

The Japanese language has various types of compound functional expressions, which are very important for recognizing the syntactic structures of Japanese sentences and for understanding their semantic contents. In this paper, we formalized the task of identifying Japanese compound functional expressions in a text as a chunking problem. We applied a machine learning technique to this task, where we employed that of Support Vector Machines (SVMs). We showed that the proposed method significantly outperforms existing Japanese text processing tools. 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 manually create and maintain those rules. On the other hand, in our framework based on the machine learning technique, it is sufficient to collect and manually annotate about 50 training examples per expression.

References


