A Hybrid Approach for Functional Expression Identification in a Japanese Reading Assistant

Gregory Hazelbeck
Graduate School of
Science and Technology
Keio University
gregh@nak.ics.keio.ac.jp

Hiroaki Saito
Graduate School of
Science and Technology
Keio University
hxs@ics.keio.ac.jp

Abstract

In this paper we present a hybrid approach for identifying Japanese functional expressions and its application in a Japanese reading assistant. We combine the results of machine learning and pattern-based methods to identify several types of functional expressions. We show that by using our approach we can double the coverage of previous approaches and still maintain the high level of performance necessary for our application.

1 Introduction

Functional expressions are one of the most important elements of Japanese grammar that anyone studying the language must learn. Despite the importance of functional expressions, many tools that assist learners of Japanese with reading texts only provide dictionary look-up of simple words. However, the ability to quickly obtain information about such grammar could not only improve the learner's comprehension of the text, but also facilitate their learning process of new elements of Japanese grammar. Thus, we have decided to develop a Japanese reading assistant that is capable of providing explanations of functional expressions in addition to vocabulary.

Functional expressions in Japanese are compound expressions that contain content and function words, and can have both compositional and non-compositional meanings. For example, in Table 1, sentences 1 and 2 contain the にあたり (niatari) compound expression. In sentence 1, this expression has a functional, non-compositional meaning of "when." However, in sentence 2, the same expression has a compositional meaning that results simply from using the post-particle に (ni) and verb あたり (a conjugated form of あたる (ataru), meaning "to hit") together. We refer to this as the content usage of a functional expression. However, there are also functional expressions where this type of content usage is very rare (or even nonexistent). Sentence 3 shows an example of the なければなりません (nakerebanarimasen) functional expression which has a very common functional meaning of "must or have to."

Tsuchiya et al. (2006) have proposed a method based on machine learning to identify functional expressions. However, this method only covers functional expressions which have balanced functional vs. content usage ratios. In order to boost coverage of current methods, we propose a hybrid approach for functional expression identification which uses a combination of the machine learning method proposed by Tsuchiya et al. (2006) and simple patterns. Coverage analysis and empirical evaluations show that our method doubles the coverage of previous approaches while still maintaining a high level of performance.

2 Related Work

2.1 Functional Expressions

Research on Japanese functional expressions has included work on identification methods as well as resources that aid identification. Matsuyoshi et al. (2006) developed a hierarchical dictionary of functional expressions called Tsutsuji. The top level of the dictionary's nine level hierarchy contains the lexical form of 341 expressions. The second level categorizes these expressions by meaning. The remaining seven levels contain various

にあたり (niatari)		
1.	Functional	アパートに入居する**にあたり**、隣近所に挨拶回りをするのは日本の習慣です。 It is a custom in Japan to greet your neighbors **when** you move into a new apartment.
2.	Content	ボールが顔面**にあたり**、歯が折れた。 The ball **hit** me in the face and broke my tooth.
なければなりません (nakerebanarimasen)		
3.	Functional	明日は学校に行か**なければなりません**。 I **have to** go to school tomorrow.

Table 1. Examples of Japanese functional expressions.

surface forms for each expression where insertion/deletion of particles and other conjugations have been made. While this is the most comprehensive dictionary of Japanese functional expressions, it can not be directly used for identification because of the functional/content usage problem described in the previous section. Therefore, identification methods like Tsuchiya et al. (2006) which uses Support Vector Machines(SVM) have been proposed to solve this problem. The data set (Tsuchiya et al., 2005) used to train this method, called MUST, contains annotated instances of 337 functional expressions. For each expression, a maximum of 50 instances were collected from the 1995 Mainichi newspaper corpus.

Recent work by the same group of researchers (Nagasaka et al., 2010) indicates that they have continued to annotate additional functional expressions for the MUST data set. During this process, they have observed that only around one third of all functional expressions possess a sufficient amount of functional and content usages to be used with their machine learning method. However, they have yet to propose any method to cover the other two-thirds of functional expressions. Our hybrid approach aims to improve coverage by identifying functional expressions that fall into this group.

3 Identification Method

Our hybrid approach combines the results from two different methods of functional expression identification. First, we will describe our implementation of a method that uses machine learning. Then, we will describe our method of generating patterns for functional expressions.

3.1 Machine Learning

Our implementation of the method proposed by Tsuchiya et al. (2006) only deviates slightly from its original form. We developed our own SVM-based text chunking system in Python while the original paper uses a text chunker called Yamcha[1]. We also use the MeCab[2] morphological analyzer with a dictionary called UniDic while the original paper used ChaSen with the default dictionary.

When training the SVMs, the original method uses three sets of labels: functional, content, and other. This allows both functional and content usages to be explicitly identified. However, in our application, we only need to identify functional usages so that the expressions' correct definitions can be displayed. Therefore, in our implementation we only use two sets of labels (functional and other) and label all content usages as other. We also decided to build a separate model for each functional expression because it enables us to add new training data and functional expressions without having to retrain everything. Although this does increase time complexity in the worse case, in practice it does not have a big affect on performance because only a small fraction of the total number of models are being used for a given text. Identification of functional expressions in a new text is performed in the following steps:

1. Morphologically analyze the text with MeCab and extract candidate functional expressions from the morpheme sequence.

2. Select the model corresponding to each candidate functional expression.

[1]http://chasen.org/~taku/software/yamcha/
[2]http://mecab.sourceforge.net/

```
GeneratePatterns($\mathcal{C}$: list of candidates from Tsutsuji)
$\mathcal{P} = \{\}$
for each candidate $c$ in $\mathcal{C}$:
  $\mathcal{S}$ = sentences that contain $c$ in the BCCWJ
  for each sentence $s$ in $\mathcal{S}$:
    $M_s$ = morpheme sequence of $s$
    $M_c$ = ExtractCandMorph($c$, $M_s$)
    if $M_c \neq null \wedge$ VerbChk($c$, $M_c$, $M_s$, $\mathcal{P}$):
      Add $M_c$ to $\mathcal{P}$
      break out of loop on line 4
    end if
  end for
end for
return $\mathcal{P}$
```

Figure 1. The GeneratePatterns algorithm.

3. Use each model to conduct chunking. Label any functional chunks as the model's corresponding functional expression.

4. Combine the results from each model. Resolve any overlapping chunks by the same rules[3] that Tsuchiya et al. (2006) use to resolve overlapping candidate functional expressions during feature generation.

3.2 Patterns

We generate simple patterns to identify functional expressions with a high ratio of functional usage. First, surveys are conducted of functional expressions in Tsutsuji using the Balanced Corpus of Contemporary Written Japanese (BCCWJ)[4]. As of writing this paper, we have selected 36 functional expressions from Tsutsuji's top level as candidates for pattern generation. We also included various surface forms of these expressions from other levels of Tsutsuji resulting in a total of 1558 candidate functional expressions. The algorithm used to generate patterns is shown in Figure 1.

The **ExtractCandMorph** function simply returns the candidate c's morpheme sequence. If the candidate's string does not match the boundaries of morphemes in M_s then *null* is returned. The **VerbChk** function returns true if a candidate is an auxiliary verb from Tsutsuji's top level and the morpheme immediately preceding it in M_s is a verb. It returns true for lower level auxiliary verb candidates if the last morpheme in its morpheme sequence is also in the morpheme sequence of its top-level parent candidate from Tsutsuji. For any candidate that is not an auxiliary verb, the function always returns true. We force candidates from lower levels to satisfy an extra condition because their lower frequency in the BCCWJ increases the probability that a sentence with the wrong expression/usage will be selected. This algorithm produces one pattern per functional expression. Each pattern is composed of the expression's morpheme sequence. This is a list where each element contains a morpheme's surface form, part of speech, and lexical form. Patterns for auxiliary verbs also check if the previous morpheme is a verb. Using this algorithm, we were able to generate 502 patterns with our 1558 candidate functional expressions.

4 Coverage Analysis

To investigate the improvement in coverage achieved by our hybrid approach, we compared the coverage of our approach with the coverage of just the MUST data set. We define coverage as the ratio of functional expressions contained in both the Tsutsuji dictionary and BCCWJ that are supported.

We first collected all of the functional expression surface forms contained in Tsutsuji. We excluded all of the single character surface forms which are mostly simple particles. Next, we recorded the frequency of each surface form's string in the BCCWJ. Overlapping of strings is allowed as long as a string covers at least one character that no other string does. Finally, we recorded which surface forms were supported by our hybrid approach and the MUST data set. Table 3 shows our final results.

Our results show that MUST is only covering around 12% of Tsutsuji's functional expressions in the BCCWJ. The additional functional expressions supported by our hybrid approach helps boost this coverage to 24%. Improvement in coverage is observed at every frequency interval. This is especially advantageous for our application because it allows us to display information about many different common and uncommon functional expressions.

[3]Specifically, select the candidate that starts at the left-most morpheme. If more than one candidate starts at the same morpheme then select the longest candidate.

[4]Balanced Corpus of Contemporary Written Japanese Monitor Release Data (2009 Version).

Corpus	**Usage Examples**		**Total Examples**	**Total Morphemes**
	Functional	**Content**		
Training (MUST)	1,767	1,463	3,230	114,699
Testing (1995 Mainichi Newspaper)	5,347	1,418	6,765	244,324

Table 2. Training and testing corpora details.

Frequency Interval	**Tsutsuji**	**MUST**		**Hybrid**	
>5,000	199	44	(22%)	70	(35%)
5,000-1,001	244	70	(29%)	111	(45%)
1,000-501	134	37	(28%)	54	(40%)
500-101	519	124	(24%)	191	(37%)
100-51	269	53	(20%)	90	(33%)
50-26	327	54	(17%)	97	(30%)
25-11	467	46	(10%)	113	(24%)
10-2	1,180	55	(5%)	188	(16%)
1	723	11	(2%)	82	(11%)
Total	4,062	494	(12%)	996	(24%)

Table 3. Functional expressions covered by each resource. Percentage of Tsutsuji covered in each frequency interval is given in parenthesis.

Software (kernel)	**Precision**	**Recall**	$F_{\beta=1}$
Yamcha (polynomial)	0.928	**0.936**	0.932
Our chunker (linear)	**0.931**	0.935	**0.933**

Table 4. Experiment 1 results.

5 Evaluation

We evaluated the machine learning method on 54 of the most difficult to identify functional expressions. These are the same expressions that were used in Tsuchiya et al. (2006)'s evaluation. Details of the training and testing data sets are shown in Table 2. Results (Table 4) show that this method performs well even on the most difficult functional expressions. We also found that using a simple linear kernel gave the best precision.

We evaluated the patterns generated from our method by using them to identify functional expressions in randomly selected texts from the BCCWJ. After verifying 2000 instances of identified functional expressions, we only found 6 instances to be incorrect. However, since these 2000 instances only cover 89 of the 502 expressions that we support, we randomly selected two instances of each remaining expression from the BCCWJ and verified them. In the additional 750 instances that were verified, only 10 instances were found to be incorrect. Results of the second experiment show that patterns generated for high frequency functional expressions are providing especially good performance.

6 Conclusion

In this paper we presented a hybrid approach for identifying Japanese functional expressions and its application in a Japanese reading assistant. We showed that a combination of machine learning and simple patterns can improve coverage while still maintaining the high level of performance necessary for our application.

7 Acknowledgements

We would like to thank Takehito Utsuro for allowing us to use his annotated functional expression data to evaluate our approach. We would also like to thank all of the other people involved in creating the MUST data set. Finally, we would like to thank the anonymous reviewers for all of their helpful comments.

References

Matsuyoshi, Suguru, Satoshi Sato, and Takehito Utsuro. 2006. Compilation of a Dictionary of Japanese Functional Expressions with Hierarchical Organization. *ICCPOL*. pp. 395–402.

Nagasaka, Taiji, Takehito Utsuro, Suguru Matsuyoshi, Masatoshi Tsuchiya. 2010. Analysis and Detection of Japanese Functional Expressions based on a Hierarchical Lexicon. *Proceedings of the 16th Annual Meeting of the Association for Natural Language Processing*. pp. 970–973. (in Japanese)

Tsuchiya, Masatoshi, Takao Shime, Toshihiro Takagi, Takehito Utsuro, Kiyotaka Uchimoto, Suguru Matsuyoshi, Satoshi Sato, and Seiichi Nakagawa. 2006. Chunking Japanese Compound Functional Expressions by Machine Learning. *Proceedings of the 2nd International Workshop on Web as Corpus (EACL-2006)*. pp. 11–18.

Tsuchiya, Masatoshi, Takehito Utsuro, Suguru Matsuyoshi, Satoshi Sato, and Seiichi Nakagawa. 2005. A Corpus for Classifying Usages of Japanese Compound Functional Expressions. *PACLING*. pp. 345–350.