Discovery of Definition Patterns by Compressing Dictionary Sentences

Masatoshi Tsuchiya[†], **Sadao Kurohashi**[‡], **Satoshi Sato**[†]
tsuchiya@pine.kuee.kyoto-u.ac.jp, **kuro@kc.t.u-tokyo.ac.jp**, **sato@i.kyoto-u.ac.jp**
[†]Graduate School of Informatics, Kyoto University
Sakyo, Kyoto, 606-8501, JAPAN
[‡]Graduate School of Information Science and Technology, Tokyo University
7-3-1 Hongou, Bunkyo, Tokyo, 113-8656, JAPAN

Abstract

This paper proposes an automatic method to discover definition patterns from a dictionary made for humans. It contains frequent patterns to describe words and concepts. Each definition pattern gives a set of similar words and can be used as a template to clarify distinctions among them. To discover these definition patterns, we convert definition sentences into tree structures, and compress them using the MDL principle. An experiment on a Japanese children dictionary is reported, showing the effectiveness of our method.

1 Introduction

How to handle the meanings of words is the first, crucial problem in realizing intelligent natural language processing. A thesaurus is one way of representing meanings of words by describing their hyponym-hypernym relations.

A thesaurus, however, lacks information about distinctions among synonyms. When 'a swimming pool' and 'an athletics stadium' are classified into the same group 'a stadium', this classification clearly shows the similarity among them, but gives no information about their distinctions. Therefore, we cannot infer the rule to distinguish between the natural sentence "A boy swims in a swimming pool" and the unnatural sentence "A boy swims in an athletics stadium". When a thesaurus which keeps distinctions among synonyms is available, it is easy to solve this difficulty.

This paper proposes an automatic method to discover sets of similar words and to find distinctions among them from an ordinary dictionary. This is surely a step building a thesaurus which can distinguish among synonyms. There are frequently recurring sub-sentential patterns in a dictionary because its definition sentences are written carefully to describe words and concepts. To discover such sub-sentential patterns, we employ data compression using the MDL principle, which is widely used to discover common substructures in given data. Our approach consists of two parts: 1) parsing definition sentences to convert them into trees, and 2) compressing the trees using the MDL principle to extract those patterns. Each extracted sub-sentential pattern gives a set of similar words and can be used as a template to clarify distinctions among them.

We now proceed as follows. In Section 2, we discuss frequent sub-sentential patterns in dictionary sentences. The next section (Section 3) explains the description length formulation of the dictionary and presents an algorithm to compress it. The experiment on a Japanese children dictionary is reported in Section 4, showing the effectiveness of our method in Section 5. We finish this paper with a conclusion section (Section 6).

2 Definition Sentences and Definition Patterns

2.1 Definition Patterns

Reading a dictionary, we find that some expressions are used frequently to describe

words and concepts. For example, the expression 'comes into flowers' is used 68 times in Reikai Shogaku Kokugojiten (Tadika, 1997). Three of them are:

aburana a type of plant which comes into yellow flowers in spring.

magnolia a type of plant which comes into white flowers in summer.

nemunoko 'silk tree' a type of plant which comes into red flowers in summer, forming a cluster.

The common expression, 'comes into flowers', indicates that these words denote plants which bear flowers. More careful observation of these sentences reveals that these sentences include words which represent colors of flowers and words which denote seasons. Using the variable $<$ color $>$ which denotes an adjective of color, and using the variable $<$season$>$ which denotes a noun representing a season, the common sub-sentential pattern among these sentences can be unified into the same format as follows:

comes into $<$color$>$ flowers in $<$season$>$.

In this paper, we will call such common sub-sentential patterns *definition patterns*, their fixed parts *common features*, and their variable parts *common slots*.

The fact that 'aburana', 'magnolia', and 'nemunoki' belong to the set of words which are described by this definition pattern indicates that they are similar. In other words, it is possible to get a set of similar words when a definition pattern is discovered. Distinctions among them are also extracted by different values of common slots.

2.2 Arbitrariness of Definition Patterns

We have focused on the definition pattern 'comes into $<$color$>$ flowers in $<$season$>$' in the previous section, and there are many possible definition patterns of different sizes, such as 'comes into $<$ color $>$ flower', 'comes into flowers' and so on. Similarly, there are possible patterns of different ranges of values. We can suppose the common slot $<$noun$>$ is the space which can hold any noun and the definition pattern 'comes into flowers in $<$noun$>$'.

To select an appropriate pattern from many possible ones, we employ the Minimum Description Length (MDL) principle, proposed by Rissanen (Rissanen, 1989). In the MDL principle, models are evaluated by the sum of the description length of the model and the description length of the given data based on it, and the model which gives the minimum sum is the best model to describe the given data. A previous study (Cook and Holder, 1994) employs the MDL principle to discover common sub-structures such as benzene from molecular structures, and others (Li, 1998; Bunt and Muskens, 1999) discover characters of language from a large corpus based on the MDL principle.

3 Dictionary Description Length

This section explains the definition of the description length of the dictionary used to compress its definition sentences using the MDL principle. In our approach, definition sentences are converted into tree structures, then definition patterns are mapped into sub-trees among them. We will define the description length of the dictionary, considering compression using common sub-trees.

3.1 Tree Representation of Definition Sentences

At first, each definition sentence of a dictionary is converted into a tree. Each of its vertices has a label corresponding to a content word, and each of its edges has a connection type representing a modifier-head relation between content words.

To perform such a conversion, we need language-dependent analysis. In our experiments, the Japanese morphological analyzer JUMAN, and the parser KNP were employed (Kurohashi and Nagao, 1994). Both of them can analyze definition sentences of a Japanese dictionary with a fairly satisfactory accuracy.

After the conversion of raw sentences into

trees, we believe our method is language-independent. Although our experiment was done on a Japanese dictionary, in this paper we use their English translations for explanation.

As a matter of notational convenience, we use the following formal definition of tree. A tree t which has n vertices is defined as a 4-tuple,

$$t = (n, V_t, E_t, \Phi_t), \tag{1}$$

where V_t denotes its vertex set, E_t denotes its edge set, and Φ_t denotes a mapping from E_t to a set of ordered pairs of instances of V_t.

3.2 Tree Representation of Definition Patterns

We focus attention on the tree representation of definition patterns included in definition sentences.

Figure 1–a illustrates how the expression of the common feature 'comes into flowers' is converted into the common sub-tree which is discovered from both trees.

Expressions in common slots, such as 'yellow' and 'white', are mapped to vertices which have an edge connecting to the common sub-tree. To represent a vertex corresponding to a common slot, we use *semantic classes* which are sets of labels of vertices corresponding to common slots. In our approach, a label is a content word and a semantic class is defined as a set of content words. Suppose the semantic class $<\text{color}> = \{\text{yellow}, \text{white}, \ldots\}$, and the definition pattern 'comes into $<\text{color}>$ flowers' can be extracted as the sub-tree B which includes a semantic class vertex in Figure 1–c.

We also introduce *connection classes* which enable more flexible expressions of definition patterns. A connection class is defined as a set of connection types, considering a hierarchy of relations between content words based on syntactic knowledge.

3.3 Formal Definition of Description Length of a Dictionary

In this section, we give a formal definition of the description length of a dictionary.

3.3.1 Class of a Target Tree Set

To employ the MDL principle, it is necessary that a class of the target data be a finite set or a countably infinite set. We introduce a label set Σ and a connection type set Γ to express a class of a target tree set which represents a dictionary. A semantic class is defined as a set of content words, and the label set Σ satisfies

$$\Sigma \subseteq 2^{\Sigma_0},$$

where Σ_0 denotes a set of all content words in the dictionary. For example, the minimum label set $\Sigma_{\min}$ to describe all trees and sub-trees in Figure 1–c is defined as

$$\Sigma_{\min} = \left\{ \begin{array}{l} \text{a type, plant, comes, flowers, spring,} \\ \text{summer, white, yellow, } <\text{color}> \end{array} \right\}.$$

Because a connection class is defined as a set of connection types, a connection type set Γ satisfies

$$\Gamma \subseteq 2^{\Gamma_0},$$

where Γ_0 denotes the set of all modifier-head relations between content words in the dictionary. Finally, considering that both the label set and the connection type set are finite sets, and that an element of the target set is a tree which is defined as in Formula 1, in follows that the class of the target tree set represents a countably infinite set.

3.3.2 Description Length of a Simple Tree

Let us first consider how to calculate the simple tree description length $L(t|\Sigma_0, \Gamma_0)$ where t denotes a discrete tree which meets these restrictions: 1) all vertices are labeled by instances of the label set Σ_0 which includes no semantic classes, 2) all edges are typed by instances of the connection type set Γ_0 which includes no type classes. Given the label set Σ_0 and the type set Γ_0, we need

$$L(t|\Sigma_0, \Gamma_0) = -\log P(t|\Sigma_0, \Gamma_0), \tag{2}$$

to encode t using a non-redundant code. When all probability distributions of size of tree, vertices, edges, and tree structure are

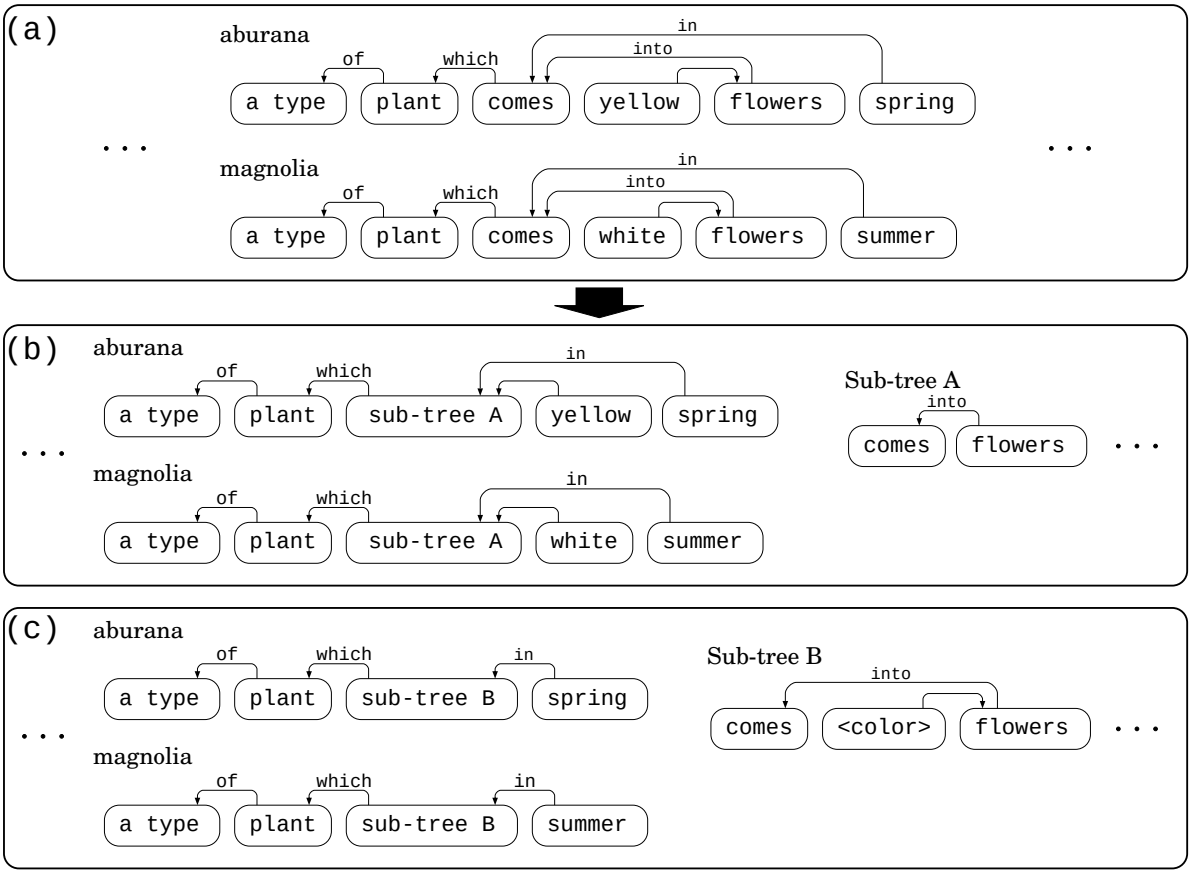

Figure 1: Compression examples of a dictionary.

independent, the probability of realizing t is defined as a joint probability distribution:

$$\begin{aligned} & P(t|\Sigma_0, \Gamma_0) \\ &= P(n, V_t, E_t, \Phi_t|\Sigma_0, \Gamma_0) \\ &= P(n)P(V_t|\Sigma_0)P(E_t|\Gamma_0)P(\Phi_t|V_t, E_t) \end{aligned} \quad (3)$$

If we assume that each occurrence of vertices is independent, the conditional probability $P(V_t|\Sigma_0)$ is estimated as

$$P(V_t|\Sigma_0) = \prod_{v \in V_t} P(v|\Sigma_0). \quad (4)$$

When the probability distribution over Σ_0 is assumed to be a uniform distribution, the conditional probability $P(v|\Sigma_0)$ is estimated as

$$P(v|\Sigma_0) = \frac{1}{|\Sigma_0|}. \quad (5)$$

Assuming that each occurrence of edges is independent, the conditional probability $P(E_t|\Gamma_0)$ of mapping from each edge of E_t to a instance of Γ_0 is estimated as

$$P(E_t|\Gamma_0) = \prod_{e \in E_t} P(e|\Gamma_0). \quad (6)$$

Assuming further that the probability distribution over Γ_0 is uniform, the conditional probability $P(e|\Gamma_0)$ is estimated as

$$P(e|\Gamma_0) = \frac{1}{|\Gamma_0|}. \quad (7)$$

In Japanese, all vertices except the last one of the sentence have an edge connecting to a following vertex. Then the number of possible tree structures is equal to the factorial of $(|V_t| - 1)$. When the probability distribution of realizing t over V_t and E_t is a uniform distribution, we estimate

$$P(\Phi_t|V_t, E_t) = \frac{1}{(|V_g| - 1)!}. \quad (8)$$

The probability $P(n)$ must satisfy two restrictions: 1) it is greater than zero over all positive numbers, and 2) it satisfies $\sum_n^\infty P(n) \leq 1$. As such a probability distribution, we employ the probability P_u that gives the unary encoding of positive numbers.

$$P(n) = P_u(n) = 2^{-n} \tag{9}$$

Finally, the description length of a tree t is derived as follows:

$$\begin{aligned} L(t|\Sigma_0, \Gamma_0) &= |V_t| \log |\Sigma_0| + |E_g| \log |\Gamma_0| \\ &\quad + \sum_{i=1}^{|V_t|} \log i - \log P_u(n). \end{aligned} \tag{10}$$

3.3.3 Description Length of a Sub-Tree

The description length of a sub-tree s is derived in a similar way to Formula 10 as

$$\begin{aligned} L(s|\Sigma, \Gamma) &= |V_s| \log |\Sigma| + |E_s| \log |\Gamma| \\ &\quad + \sum_{i=1}^{|V_s|} \log i - \log P_u(n), \end{aligned} \tag{11}$$

where V_s denotes a vertex set of s, E_s denotes an edge set of s.

3.3.4 Description Length of a Condensed Tree

We turn to the description length of a tree t which was condensed by using sub-trees included in the sub-tree set Ω. Sub-trees may include semantic class vertices and connection class edges, so we require more bits to describe their actual labels and their actual types. Some bits are also required to describe possibilities of connections inside and outside of sub-trees.

Let us consider bits which are required to describe actual labels of semantic class vertices. The number of possible mappings from each vertex of the vertex set V_s to its actual label in Σ_0 is computed as

$$X(s) = \prod_{v \in V_s} |C(v)|, \tag{12}$$

where $C(v)$ is the function of v which gives its semantic class. As a matter of notational convenience, we assume that $C(v)$ returns $\{v\}$ when the label of the vertex v is not a semantic class. If we assume that the distribution of the occurrences of actual labels is uniform, Formula 5 is replaced by this expression,

$$P'(v|\Sigma) = \frac{1}{|\Sigma|} \cdot \frac{1}{X(v)}. \tag{13}$$

We will now discuss the bits required to describe the actual types of connection class edges in the similar way. On the sub-tree s, the number of possible mappings from each edge of the edge set E_s to its actual label in Γ_0 is computed as

$$Y(s) = \prod_{e \in V_s} |C(e)|, \tag{14}$$

where $C(e)$ is the function of e which gives its connection class. The function $C(e)$ returns $\{e\}$ when the type of the edge e is not connection class. If we assume that the distribution of the occurrences of actual types is uniform, Formula 7 is replaced by this expression,

$$P'(e|\Gamma) = \frac{1}{|\Gamma|} \cdot \frac{1}{Y(e)}. \tag{15}$$

When a vertex is labeled as a sub-tree and some edges connect to it, to distinguish their inside end, some bits are also required. Therefore, Formula 8 is replaced by this expression,

$$P'(\Phi_t|V_t, E_t) = \frac{1}{(|V_t| - 1)! \cdot \prod_{v \in V_t} |V_v|^{|E_v|}}, \tag{16}$$

where E_v denotes a set of edges connecting to the vertex v, V_v denotes a vertex set of the sub-tree v. When v is not labeled as a sub-tree, we set V_v to $\{v\}$.

Finally, the description length of a tree t which was condensed using sub-trees is defined as

$$\begin{aligned} &L(t|\Sigma + \Omega, \Gamma) \\ &= |V_t| \log |\Sigma + \Omega| + \sum_{v \in V_t} \log |C(v)| \\ &\quad + |E_t| \log |\Gamma| + \sum_{e \in E_t} \log |C(e)| \\ &\quad + \sum_{i=1}^{|V_t|} \log i + \sum_{v \in V_t} |E_v| \log |V_v| \\ &\quad - \log P_u(n). \end{aligned} \tag{17}$$

3.3.5 Description Length of a Dictionary

Consequently, the description length of a set of trees D is derived as

$$L(D) = \sum_{t \in D} L(t|\Sigma + \Omega, \Gamma) + \sum_{s \in \Omega} L(s|\Sigma, \Gamma) + L(\Sigma) + L(\Gamma), \quad (18)$$

where $L(\Sigma)$ denotes the description length of Σ, $L(\Gamma)$ denotes the description length of Γ.

Since there is no clear basis for selecting a good Σ, we assume that $L(\Sigma)$ is equal for all Σ. The same may be said of $L(\Gamma)$. Then, the third and fourth item of Formula 18 can be omitted, and so we obtain

$$L'(D) = \sum_{t \in D} L(t|\Sigma + \Omega, \Gamma) + \sum_{s \in \Omega} L(s|\Sigma, \Gamma) \quad (19)$$

as the objective function of the dictionary compression based on the MDL principle.

3.4 Search Algorithm

As we mentioned above, the goal of the MDL based compression is to find a label set Σ, a type set Γ and a set of sub-trees Ω which minimize the value of the objective function. The possible combination of Σ, Γ and Ω, however, becomes so large that it is intractable to find the optimal solution by considering all possible cases.

Furthermore, since they cannot be changed independently, neither the divide and conquer method nor dynamic programming can handle this problem. Therefore, we do not aim at finding the optimal solution, and take an iterative improvement method based on heuristics instead.

First of all, we set Σ to the set of all content words in the dictionary and all semantic classes in a handmade thesaurus to reduce search space. Secondly, a fixed type set Γ is prepared based on syntactic knowledge of the target language. Having discussed that it is impossible to obtain the optimal solution of a sub-tree set Ω, we design a search procedure based on an iterative improvement method for Ω and a beam search for each element of Ω. Our search procedure consists of the following steps:

1. A list of all pairs of vertices is made which exist in the target tree set D.
2. If a sub-tree in the list contains a word vertex or a simple typed edge, new sub-trees are added to the list whose word vertex or simple typed edge is replaced with all semantic classes or type classes to which the vertex and the edge belongs.
3. All sub-trees are sorted depending on their frequency. For the top-n sub-trees, their scores are calculated and the remaining sub-trees are deleted from the list. The score of a sub-tree s is equal to the value of the objective function when s is added to Ω.
4. For the best sub-tree which gives the minimum score, if its score is worse than L'_{best}, go to step 7.
5. The best sub-tree is substituted for s_{best}, and its score for L'_{best}.
6. All sub-trees in the list are sorted depending on their score. New sub-trees which cover one of the the m-best sub-trees and an additional vertex are added to the new list. The list is replaced with the new one, then the procedure is repeated from step 2.
7. The best sub-tree s_{best} is added to Ω, and each of its occurrences in D is replaced by a condensed vertex.
8. This iterative procedure will be continued while the last replacement improves the value of the objective function.

4 Experiment

We applied the method described so far to Reikai Shogaku Kokugojiten, a Japanese dictionary for children (Tadika, 1997). Table 1 shows statistics of the dictionary. Definition sentences were extracted from the dictionary by a simple filter, and converted to tree structures by JUMAN and KNP.

It was impossible to conduct an experiment by using a whole dictionary because

of the memory limit of our computer (Our computer is Sun Enterprise–3500 with 4GB memory). Therefore, we eliminated half of the head words and those sentences which include words which have no category in the thesaurus.

Table 1: Statistics of the dictionary.

	whole	target
# of head words	28015	10087
# of definition sentences	52919	13486
# of vertices	179597	53212

As a thesaurus to reduce the search space of Σ, we employed Bunruigoihyou, a Japanese thesaurus (Nat, 1993). For words which have several categories on the thesaurus, one of them was selected randomly and the others were ignored (Words of this thesaurus have 1.2 categories on average). Because most general semantic classes confuse our heuristic search procedure which depends on frequency, such classes which were close to the root of the thesaurus were deleted. Next, we defined the fixed type set Γ based on dependency-types between content words assigned by KNP.

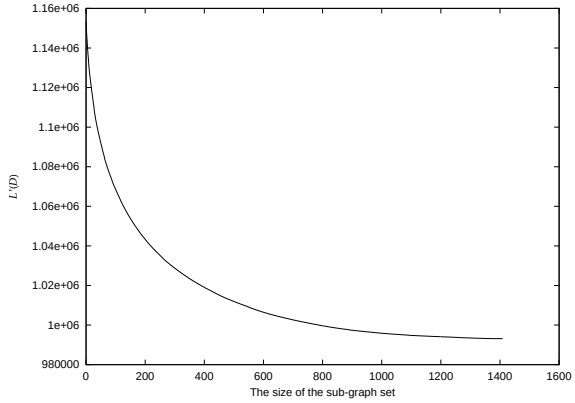

Figure 2: Description length of the dictionary.

We employed the discussed algorithm on the fixed sets Σ and Γ, and discovered 1409 sub-trees. Through this search process, the value of the objective function decreased from 1153455.5 bit to 993121.6 bit. Figure 2 shows the trace of $L'(D)$. It was equivalent to 13.9% compression.

5 Discussion

Figure 4 shows some compressed definition sentences which are translated into English. Most features detected by our method are reasonable based on our linguistic intuition.

The common sub-tree "to < arrange > neatly" is discovered from three definition sentences of *tidying*, *arrangement* and *appearance*. Definition sentences of other head words such as *hairdressing* also include the similar sub-tree "to < arrange >", but these three head words are the most similar among them. We found that there is no suitable word among 22 head words which include the expression, "neatly".

The common sub-tree "is <cook> to eat" is discovered from the definition sentences of the five head words such as *kidney beans* and *spaghetti*. All these five head words denote foodstuff and it is interesting that the proper semantic class < cook > is discovered. The definition sentences of *asari*, *eel* and *shijimi* include the different sub-tree "is made into <dish> to eat", which includes the common feature "to eat", and they also concern foodstuff. There are few foodstuff head words except these eight head words in the dictionary.

We discovered the common sub-tree "comes into < color > flowers" from the definition sentences whose head words are *gardenia*, *magnolia* and so on. These head words are hyponyms of plants which bloom flowers. These examples show the effectiveness of our method which can extract the hyponym-hypernym relation and the attribute <color> which is shared by these hyponyms.

6 Conclusion

In this paper, we proposed an automatic method to discover sets of similar words and distinctions among them. Observing definition sentences in a real dictionary written in natural language, we found that there are definition patterns which are used frequently to describe words and concepts. These definition patterns consist of common features and common slots, and give sets of similar words and distinctions among them. To discover these

haircut
to shape the hair.
<arrange>

tidying
to arrange neatly.
<arrange>

arrangement
to tidy things neatly.
<arrange>

appearance
to straighten clothes neatly.
<arrange>

precise
neatly and correct in regard to the smallest details.

kidney beans
their green pods which are boiled to eat, and
<cook>
their beans which are made into anko.

spaghetti
an Italian food which is boiled to eat and served
<cook>
with some kind of sauce.

soumen 'vermicelli'
a Japanese food which is boiled to eat.
<cook>

tanishi 'mud snail'
food snail which is poached to eat.
<cook>

hijiki 'algae'
food algae which is boiled to eat.
<cook>

asari 'clam'
food clam which is made into miso soup to eat.
<dish>

eel
food fish which is made into kabayaki to eat.
<dish>

shijimi 'clam'
food clam which is made into miso soup to eat.
<dish>

gardenia
a type of plant which comes into aromatic white
<color>
flowers in summer

magnolia
a type of plant which comes into big white
<color>
flowers in early summer

Figure 3: Samples of compressed definition sentences.

patterns, an automatic method was designed. Its first step is parsing dictionary sentences to convert them into trees, and its second step is to compress the trees using the MDL principle. We reported an experiment to compress a Japanese children's dictionary, and discovered several interesting results of definition patterns among definition sentences, indicating the effectiveness of our method.

The target of our future research is to generate a thesaurus which includes descriptions of distinctions among synonyms.

References

H. Bunt and R. Muskens, editors, 1999. *Minimum Description Length and Compositionality*, volume 1, pages 113–128. Kluwer.

Diane J. Cook and Laerence B. Holder. 1994. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1:231–255.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4).

Hang Li. 1998. Generalizing case frames using a thesaurus and the mdl principle. *Computational Linguistics*, 24(2):217–244.

The National Language Research Institute, 1993. *Bunruigoihyou*.

J. Rissanen. 1989. *Stochastic Complexity in Stochastic Inquiry*. World Scientific Publishing Company.

Junichi Tadika, editor. 1997. *Reikai Shogaku Kokkugojiten*. Sansei-do Co.