International Journal of Computational Linguistics and Applications vol. 7, no. 2, 2016, pp. 183–199
Received 05/02/2016, accepted 06/03/2016, final 13/09/2016
ISSN 0976-0962, http://ijcla.bahripublications.com

Correcting Redundant Japanese Sentences Using Patterns and Machine Learning for the Development of Writing Support Systems

MASAKI MURATA[1], SHUNSUKE TSUDO[1],
MASATO TOKUHISA[1], AND QING MA[2]

[1] *Tottori University, Japan,*
[2] *Ryukoku University, Japan*

ABSTRACT

In this study, we propose a method to automatically correct redundant sentences using patterns and machine learning and propose methods that combine pattern-based and machine-learning methods. We conducted experiments to correct redundant sentences containing "kanou" (possible or possibility), "toiu" ("that is" or called), and "surukoto" (to do). The results demonstrate that the proposed method can correct redundant sentences at an accuracy of 0.6 and estimate corrected expressions against redundant parts at an accuracy of 0.7; furthermore, we created a method to support a user's writing. In this method, a system displays redundant parts and provides candidate expressions.

KEYWORDS: *redundant sentence, pattern, machine learning, writing support system*

1 INTRODUCTION

Redundancy is a problem in sentence generation and correction [1–9]. Redundant sentences hinder readability; therefore, such sentences should be changed to make sentences simple, clear, and correct.

The following sentence is an example of redundancy.

"At the beginning, we firstly perform the check of the machine."

In this sentence, "at the beginning" and "firstly" are redundant because they indicate the same meaning. In the phrase "perform the check," "perform" is redundant and can be omitted. The sentences with words that have the same meaning as well as words that appear more than once or phrases that contain unnecessary words are redundant and confusing. The abovementioned sentence can be changed to the following simple sentence by modifying and deleting the redundant parts.

"We firstly check the machine."

In this study, we collect and analyze redundant sentences and construct a method to automatically detect redundant sentences.

The following steps can improve the quality of sentences:

- correcting typographical errors and selecting appropriate expressions
- correcting errors in word order and dependency
- modifying redundant sentences

Many studies have been conducted for correcting typographical errors and selecting appropriate expressions [3, 4, 8] and for correcting the word order and word dependency errors [5, 7]); however, few studies have investigated the modification of redundant sentences.

Note that redundant expressions are deleted in automatic summarization methods [10], and in machine translation, automatic translation systems often produce redundant expressions [6]. Thus, techniques have been developed to modify redundant sentences, and these techniques are useful for automatic summarization and machine translation.

Tsudo et al. handled the automatic detection of redundant sentences in Japanese [11]. They detected redundant sentences with slightly high F-measures (0.7–0.8); however, in this study, we handle the automatic correction of redundant sentences in Japanese.

Tsudo et al. indicated that expressions containing "*kanou*" (possible or possibility), "*toiu*" ("that is" or called), and "*surukoto*" (to do) are possibly redundant [11]; therefore, in this study, we handle the automatic correction of expressions containing "*kanou*," "*toiu*," and "*surukoto*." A method for correcting these expressions would help in constructing methods for correcting other expressions.

In our study, we employed redundant sentences as inputs and changed the input sentences to nonredundant sentences. The proposed method can support writers; moreover, it can be used to construct short sentences by reducing redundant sentences to a limited number of characters.

The characteristics of this study are described as follows.

- Our study has an originality that we handled automatical correction of redundant sentences. We proposed a method to automatically correct redundant sentences using patterns and machine learning.
- In our experiments, our proposed method could correct redundant sentences at an accuracy of 0.6 and estimate corrected expressions against redundant parts at an accuracy of 0.7.
- We created a method to support a user's writing for correcting redundant sentences. In this method, a system displays redundant parts and provides candidate expressions.

2 TASK

The system uses 100 redundant sentences (Section 3) containing "*kanou*," "*toiu*," and "*surukoto*" as inputs and changes them to nonredundant sentences. We call "*kanou*," "*toiu*," and "*surukoto*" *target words*.

We use 100 redundant sentences constructed in Section 3 as inputs. We use 100 corrected sentences constructed in Section 3 as correct sentences and the system's output is determined to be correct when it changes an input sentence into correct sentences. For evaluation, we use two-fold cross validation. In each method, the system uses the input and correct sentences as a training data set, and input sentences are used as the test data set.

3 Data

We employed Japanese Wikipedia pages (Oct. 30, 2013) for our experiments and collected sentences containing target words. We eliminated sentences that contained multiple target words and extracted 100 sentences in which an expression containing a target word can be changed into a nonredundant expression. We manually corrected 100 extracted sentences and constructed 100 pairs of extracted and corrected sentences. The abovementioned correction was performed against expressions that are redundant because of the existence of target words. Note that we generated over 100 pairs for "*kanou*," "*toiu*," and "*surukoto*" (300 pairs in total).

An example of a redundant sentence and its corrected sentence is as follows:

We show an example of a redundant sentence and its corrected sentence below.

Redundant sentence:

juubun	*rikai*	*kanou*	*dearu*
(enough)	(understand)	(possible to)	(be)

(I am possible to understand it enough.)

Corrected sentence:

juubun	*rikai*	*dekiru*
(enough)	(understand)	(can)

(I can understand it.)

4 Proposed method

We constructed five methods for automatically correcting redundant sentences. In all these methods, we independently handle each target word. Note that independent processing of each target word is often used for word sense disambiguation [12].

- Method 1: Patterns
- Method 2: Machine learning
- Method 3: Baseline
- Method 4: Stacking
- Method 5: Best method

We use a support vector machine (SVM) for machine learning.

4.1 *Method 1: Patterns*

We correct redundant sentences using patters.

In Method 1, we correct redundant sentences using patterns and detect differences between a redundant sentence and its corrected sentence using a word unit. An example of detecting differences is as follows: "*kanou dearu*" (is posibble to) and "*dekiru*" (can) are differences, whereas the following redundant sentence is given.

Redundant sentence:

juubun	*rikai*	*kanou*	*dearu*
(enough)	(understand)	(possible to)	(be)

Corrected sentence:

juubun	*rikai*	*dekiru*
(enough)	(understand)	(can)

We generate a pattern from these differences, and from the abovementioned example, we create the pattern "*kanou dearu* $\rightarrow$ *dekiru*." This pattern indicates that the system changes *kanou dearu* into *dekiru*.

Thus, we create patterns using a training data set and correct redundant sentences using these patterns.

There are some cases where multiple patterns can be used for a redundant sentence. To select a pattern among multiple patterns, we use the following two methods.

PT1 We use a pattern that matches the longest expression in a redundant sentence. When there are multiple patterns that match the longest expression, we use the most frequently occurring pattern in the training data set.

PT2 We use the most frequently occurring pattern in the training data set.

4.2 *Method 2: Machine learning*

In Method 2, we correct redundant sentences using machine learning. We use an SVM (a linear kernel function) for machine learning [13–20].

Similar to Method 1, in Method 2, we use differences and create output categories (outputs) in machine learning using these differences.

We use two methods for the output.

ML1 "X $\rightarrow$ Y" combining an original expression X and a corrected expression Y in a difference part is used as output.

ML2 A corrected expression in a difference part is used as output.

We use machine learning to estimate the abovementioned outputs from an input sentence using a training data set and estimated the outputs of an input sentence in a test data set using machine learning.

The features used in machine learning are the two words and their parts of speech (POS) that appear just before the target word, and the two words and their POS that appear just after the target word.

4.3 *Method 3: Baseline*

We use the most frequent category in the training data set as output as the baseline method. We use two methods for Method 3.

BL1 The most frequent category based on ML1 of Method 2 in the training data set is always used as output.

BL2 The most frequent category based on ML2 of Method 2 in the training data set is always used as output.

4.4 *Method 4: Stacking*

Method 4 uses a stacking algorithm [21], which adds the outputs of the other methods, into features. To take advantage of Methods 1 to 3, we add the outputs of Methods 1 to 3 to the features of Method 2.

Note that we used two methods for Method 4.

ST1 The categories of ML1 are used. The features of Method 2 and the outputs of PT1, PT2, ML1, and BL1 are used as features.

ST2 The categories of ML2 are used. The features of Method 2 and the outputs of PT1, PT2, ML2, and BL2 are used as features.

4.5 *Method 5: Best method*

The best method among Methods 1–4 in a divided data set of two-fold cross validation is used for solving the other divided data set of two-fold cross validation.

We employ two types of methods for Method 5.

BEST1 The best method among PT1, ML1, BL1, and ST1 is used.

BEST2 The best method among PT2, ML2, BL2, and ST2 is used.

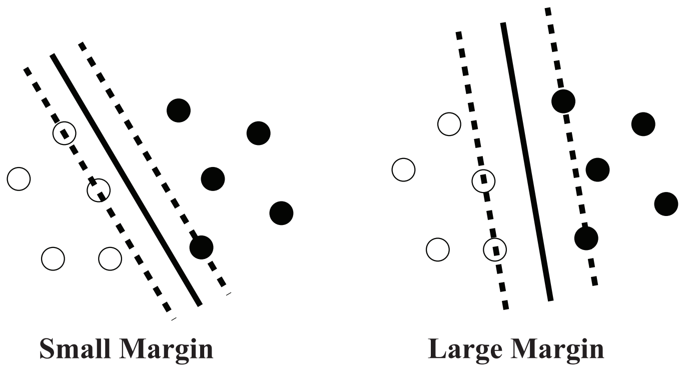


Fig. 1. Maximizing the margin

4.6 *Support vector machine method*

In this method, data comprising two categories is classified by dividing the space with a hyperplane. When the margin between examples that belong to one category and examples that belong to the other category in the training data is larger (see Figure 1[3]), the probability of incorrectly choosing categories in the open data is smaller. The hyperplane that maximizes the margin is determined

[3] In the figure, white and black circles indicate examples that belong to one category and examples that belong to the other category, respectively. The solid line indicates the hyperplane dividing the space, and the broken lines indicate planes at the boundaries of the margin regions.

and classification is performed using this hyperplane. Although the basics of this method are as described above, generally for extended versions of the method, the inner region of the margin in the training data can include a small number of examples, and the linearity of the hyperplane is changed to nonlinearity using kernel functions. Classification in the extended methods is equivalent to classification using the following discernment function, and the two categories can be classified based on whether the output value of the function is positive or negative [14, 22]:

$$\begin{aligned} f(\mathbf{x}) &= sgn\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \qquad (1)\\ b &= -\frac{max_{i,y_i=-1} b_i + min_{i,y_i=1} b_i}{2} \\ b_i &= \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i), \end{aligned}$$

where $\mathbf{x}$ is the context (a set of features) of an input example and $\mathbf{x}_i$ and $y_i (i = 1, ..., l, y_i \in \{1, -1\})$ indicate the context of the training data and its category, respectively. The function sgn is defined as follows:

$$\begin{aligned} sgn(x) = \quad & 1 \quad (x \geq 0), \qquad (2)\\ & -1 \quad (otherwise). \end{aligned}$$

Each $\alpha_i (i = 1, 2...)$ is fixed when the value of $L(\alpha)$ in Equation (3) is maximum under the conditions of Equations (4) and (5).

$$L(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j}) \qquad (3)$$

$$0 \leq \alpha_i \leq C \ (i = 1, ..., l) \qquad (4)$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{5}$$

Although the function K is called a kernel function, and various types of kernel functions can be used, in this study, we use the following polynomial function:

$$K(\mathbf{x}, \mathbf{y}) \quad = (\mathbf{x} \cdot \mathbf{y} + 1)^d, \tag{6}$$

where C and d are constants set by experimentation. In this study, C and d are fixed at 1 and 1, respectively, for all experiments. A set of $\mathbf{x}_i$ that satisfies $\alpha_i > 0$ is called a support vector, and the portion used to perform the sum in Equation (1) is calculated using only examples that are support vectors.

SVM methods can handle data comprising two categories. Generally, data comprising more than two categories can be handled using the pair-wise method. In this method, for data comprising N categories, all pairs of two different categories (N(N-1)/2 pairs) are constructed. Better categories are determined using a two-category classifier (in this study, an SVM[4] is used as the two-category classifier.). Finally, the correct category is determined by "voting" on the N(N-1)/2 pairs analyzed by the two-category classifier.

The SVM method used in this study is implemented by combining the SVM method and the pair-wise method.

5 Experiments

We conducted experiments using the methods described in Section 4.

The pair of original and corrected expressions for each target word used in BL1 (most frequent pair) is as follows.

[4] TinySVM [22] developed by Kudoh was used as the SVM.

Table 1. Accuracy rates of correctly estimating original and corrected expressions

Method	*kanou*	*toiu*	*surukoto*	Total
PT1	0.64(64/100)	0.56(56/100)	0.32(32/100)	0.51(152/300)
PT2	0.58(58/100)	0.66(66/100)	0.55(55/100)	0.60(179/300)
ML1	0.56(56/100)	0.75(75/100)	0.56(56/100)	0.62(187/300)
BL1	0.26(26/100)	0.66(66/100)	0.33(33/100)	0.42(125/300)
ST1	0.56(56/100)	0.76(76/100)	0.59(59/100)	0.64(191/300)
BEST1	0.57(57/100)	0.70(70/100)	0.59(59/100)	0.62(186/300)

Target word	Original expression	Corrected expression
kanou	*kanou dearu* (be possible to)	*dekiru* (can)
toiu	*toiu* (that is)	ϕ
surukoto	*surukoto* (to do)	ϕ

The corrected expression for each target word used in BL2 (most frequent corrected expression) is as follows.

Target word	Corrected expression
kanou	*dekiru*
toiu	ϕ
surukoto	ϕ

ϕ indicates that the system deletes an original expression.

The accuracy rates of correctly estimating original and corrected expressions are shown in Table 1. The accuracy rates of correctly estimating only a corrected expression are shown in Table 2.

From Tables 1 and 2, ML1, ML2, ST1, ST2, BEST1, and BEST2 related to machine learning obtained higher accuracy than the other methods. These methods demonstrated accuracy of approximately 0.6 for estimating the original and corrected expressions (Table 1)

Table 2. Accuracy rates of correctly estimating only a corrected expression

Method	*kanou*	*toiu*	*surukoto*	Total
PT1	0.73(73/100)	0.80(80/100)	0.33(33/100)	0.62(186/300)
PT2	0.60(60/100)	0.80(80/100)	0.57(57/100)	0.66(197/300)
ML2	0.65(65/100)	0.81(81/100)	0.65(65/100)	0.70(211/300)
BL2	0.67(67/100)	0.80(80/100)	0.43(43/100)	0.63(190/300)
ST2	0.65(65/100)	0.82(82/100)	0.63(63/100)	0.70(210/300)
BEST2	0.73(73/100)	0.76(76/100)	0.65(65/100)	0.72(216/300)

and accuracy of approximately 0.7 for estimating corrected expressions (Table 2).

Examples of sentences that were properly corrected using machine learning (ML1) are shown in Table 3.

6 Outputting hints for correction

We could correct redundant sentences at an accuracy of 0.6 and estimate corrected expressions at an accuracy of 0.7 for "*kanou*," "*toiu*," and "*surukoto*." However, a more precise method is necessary to correct redundant sentences in actual document arrangement and modification.

We could consider a method in which the system does not correct sentences but automatically detect parts that require correction to show the user candidate correct expressions for redundant parts by order of frequency (Figure 2). Note that a technique proposed in a previous study [11] can be used to detect redundant parts. We could consider another method in which the system shows candidate correct expressions by order of probability because machine learning can output probabilities that represent the degree of correctness of a candidate. With such methods, the cost of manually correcting documents would be decreased because redundant parts and their candidates for correction are displayed.

Table 3. Properly corrected examples

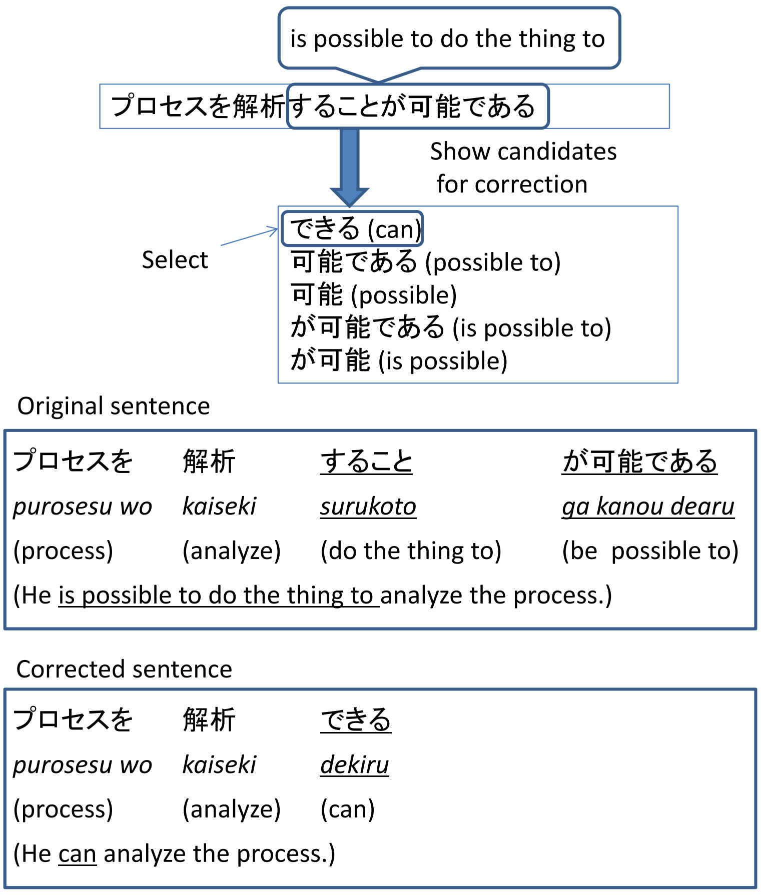
Original expression (*kanou*)
kono kangaekata wa tekiou kanou dearu
(this idea) (be applied) (possible) (be)
(This idea is possible to be applied.)
Corrected expression
kono kangaekata wa tekiou dekiru
(this idea) (be applied) (can)
(This idea can be applied.)

Original expression (*toiu*)
nagasa ga keisoku sareru toiu koto ga wakwaru
(length) (be measured) (that is) (thing) (we find)
(We find the thing that is that a length is measured.)
Corrected expression
nagasa ga keisoku sareru koto ga wakwaru
(length) (be measured) (that) (we find)
(We find that a length is measured.)

Original expression (*surukoto*)
sekibun wo keisan surukoto ga dekita
(integral) (calculate) (do the thing to) (we could)
(We could do the thing to calculate integral.)
Corrected expression
sekibun wo keisan dekita
(integral) (calculate) (we could)
(We could calculate integral.)

7 Conclusion

We have proposed a method to automatically correct redundant sentences using patterns and machine learning and proposed methods that combine pattern-based and machine-learning methods. We performed experiments to correct redundant sentences containing "*kanou*" (possible or possibility), "*toiu*" ("that is" or called), and "*surukoto*" (to do). In the experiments, the proposed method could correct redundant sentences at an accuracy of 0.6 and estimate corrected expressions against redundant parts at an accuracy of 0.7;


is possible to do the thing to
プロセスを解析することが可能である
Show candidates for correction
Select
できる (can)
可能である (possible to)
可能 (possible)
が可能である (is possible to)
が可能 (is possible)

Original sentence

プロセスを	解析	<u>すること</u>	<u>が可能である</u>
purosesu wo	*kaiseki*	*<u>surukoto</u>*	*<u>ga kanou dearu</u>*
(process)	(analyze)	(do the thing to)	(be possible to)

(He <u>is possible to do the thing to</u> analyze the process.)

Corrected sentence

プロセスを	解析	<u>できる</u>
purosesu wo	*kaiseki*	*<u>dekiru</u>*
(process)	(analyze)	(can)

(He <u>can</u> analyze the process.)

Fig. 2. Example of outputting hints for correction

moreover, we created a method to support a user's writing. In this method, the system shows the user redundant parts and candidate expressions. We believe that the proposed methods will be useful for developing writing support systems.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 26330252.

REFERENCES

1. Humphreys, K., Calcagno, M., Weise, D.: Reusing a statistical language model for generation. In: Proceedings of the EWNLG. (2001) 1–6
2. Kevin Knight, V.H.: Two-level, many-paths generation. In: Proceedings of the ACL. (1995) 252–260
3. Knight, K., Chander, I.: Automatic postediting of documents. In: Proceedings of the National Conference on Artificial Intelligence (AAAI). (1994) 779–784
4. Murata, M., Isahara, H.: Automatic detection of mis-spelled Japanese expressions using a new method for automatic extraction of negative examples based on positive examples. IEICE Transactions on Information and Systems **E85–D**(9) (2002) 1416–1424
5. Murata, M., Uchimoto, K., Ma, Q., Isahara, H.: Magical number seven plus or minus two: Syntactic structure recognition in Japanese and English sentences. In: Computational Linguistics and Intelligent Text Processing, Second International Conference, CICLing 2001, Mexico City, February 2001 Proceedings. (2001) 43–52
6. Ogino, S., Nasukawa, T.: Generating natural sentences by using shallow discourse information. In: Proceedings of the TMI. (1997) 39–46
7. Uchimoto, K., Murata, M., Ma, Q., Sekine, S., Isahara, H.: Word order acquisition from corpora. In: Proceedings of the COLING '2000. (2000)
8. Uchimoto, K., Sekine, S., Isahara, H.: Text generation from keyword. In: Proceedings of the COLING '2002. (2002) 1037–1043
9. Vestre, E.J.: An algorithm for generating non-redundant quantifier scopings. In: Proceedings of the EACL. (1991) 251–256
10. Hendrickx, I., Daelemans, W., Marsi, E., Krahmer, E.: Reducing redundancy in multi-document summarization using lexical semantic similarity. In: Proceedings of the 2009 Workshop on Language Generation and Summarisation, ACL-IJCNLP 2009. (2009) 63–66
11. Tsudo, S., Murata, M., Tokuhisa, M., Ma, Q.: Machine learning for analysis and detection of redundant sentences toward development of writing support systems. In: Proceedings of the 13th International Symposium on Advanced Intelligent Systems. (2012) 2225–2228
12. Murata, M., Utiyama, M., Uchimoto, K., Ma, Q., Isahara, H.: Japanese word sense disambiguation using the simple bayes and support vector machine methods. In: Proceedings of the SENSEVAL-2. (2001)

13. Chen, P., Wen, T.: Margin maximization model of text classification based on support vector machines. Machine Learning and Cybernetics (2006) 3514–3518
14. Cristianini, N., Shawe-Taylor, J.: An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press (2000)
15. Isozaki, H., Kazawa, H.: Efficient support vector classifiers for named entity recognition. In: Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002). (2002) 1–7
16. Kudoh, T., Matsumoto, Y.: Use of support vector learning for chunk identification. In: Proceedings of the CoNLL-2000. (2000) 142–144
17. Mitsumori, T., Fation, S., Murata, M., Doi, K., Doi, H.: Gene/protein name recognition based on support vector machine using dictionary as features. BMC Bioinformatics **6(Suppl 1)**(S8) (2005) 1–10
18. Murata, M., Ma, Q., Isahara, H.: Comparison of three machine-learning methods for Thai part-of-speech tagging. ACM Transactions on Asian Language Information Processing **1**(2) (2002) 145–158
19. Murata, M., Tanji, H., Yamamoto, K., Saeger, S.D., Kakizawa, Y., Torisawa, K.: Extraction from the web of articles describing problems, their solutions, and their causes. IEICE Transactions on Information and Systems **E94–D**(3) (2011) 734–737
20. Takeuchi, K., Collier, N.: Bio-medical entity extraction using support vector machine. In: Proceedings of the ACL 2003 Workshop on NLP in Biomedicine. (2003) 57–64
21. Halteren, H.V., Zavrel, J., Daelemans, W.: Improving accuracy in word class tagging through the combination of machine learning systems. Computational Linguistics **27**(2) (2001) 199–229
22. Kudoh, T.: TinySVM: Support Vector Machines. http://cl.aist-nara.ac.jp/˜taku-ku//software/TinySVM/index.html (2000)

Masaki Murata
Tottori University,
4-101 Koyama-Minami, Tottori 680-8552, Japan
E-mail: <murata@ike.tottori-u.ac.jp>

Shunsuke Tsudo
Tottori University,
4-101 Koyama-Minami, Tottori 680-8552, Japan
E-mail: <s082034@ike.tottori-u.ac.jp>

MASATO TOKUHISA
TOTTORI UNIVERSITY,
4-101 KOYAMA-MINAMI, TOTTORI 680-8552, JAPAN
E-MAIL: <TOKUHISA@EECS.TOTTORI-U.AC.JP>

QING MA
RYUKOKU UNIVERSITY,
SETA, OTSU, SHIGA 520-2194, JAPAN
E-MAIL: <QMA@MATH.RYUKOKU.AC.JP>