A Supervised Multiclass Classifier for an Autocoding System

Yukako TOKO (ytoko@nstac.go.jp)
National Statistics Center, Research and Development Division, Japan

Kazumi WADA (kwada@nstac.go.jp)
National Statistics Center, Research and Development Division, Japan

Mariko KAWANO (mkawano@nstac.go.jp)
National Statistics Center, Research and Development Division, Japan

ABSTRACT

Classification is often required in various contexts, including in the field of official statistics. In the previous study, we have developed a multiclass classifier that can classify short text descriptions with high accuracy. The algorithm borrows the concept of the naïve Bayes classifier and is so simple that its structure is easily understandable. The proposed classifier has the following two advantages. First, the processing times for both learning and classifying are extremely practical. Second, the proposed classifier yields high-accuracy results for a large portion of a dataset. We have previously developed an autocoding system for the Family Income and Expenditure Survey *in Japan that has a better performing classifier. While the original system was developed in Perl in order to improve the efficiency of the coding process of short Japanese texts, the proposed system is implemented in the R programming language in order to explore versatility and is modified to make the system easily applicable to English text descriptions, in consideration of the increasing number of R users in the field of official statistics. We are planning to publish the proposed classifier as an R-package. The proposed classifier would be generally applicable to other classification tasks including coding activities in the field of official statistics, and it would contribute greatly to improving their efficiency.*

Keywords: *Coding, Text classification, Naïve Bayes, Machine learning*
JEL Classification: *C38*

1. INTRODUCTION

We have developed a simple multiclass classifier that can classify short text descriptions with high accuracy (Toko *et al*., 2017; Wada and Toko, 2017; Shimono *et al.*, 2017). The proposed classifier is a supervised classifier that employs the concept of the naïve Bayes classifier. The advantages of the proposed classifier are as follows.

- The proposed algorithm is white-box algorithm and is so simple that its structure is easily understandable.
- The processing times for both learning and classification are short.
- The proposed algorithm provides high-accuracy results for a large portion of a dataset.

Coding tasks often appear in the process of survey tabulation in the field of official statistics. Coding involves matching a code for a given classification to textual descriptions. For example, since free-description fields, such as fields related to occupation, industry, family income, and expenditures, and activities for time use, sometimes appear in survey questionnaires, we are required to assign matched codes to these data for the tabulation process.

Coding studies have been conducted by a number of national institutes of statistics. Hacking and Willenborg (2012) described the study of coding activities, including autocoding systems for governmental surveys in the Netherlands. Gweon *et al*. (2017) proposed three methods for automated occupation coding, which are based primarily on statistical learning. However, there have been no studies in this field on the use of a simplified naïve Bayes classifier such as we are proposing.

We previously developed an autocoding system for the *Family Income and Expenditure Survey* in Japan (Toko *et al*., 2017; Wada and Toko, 2017; Shimono *et al.*, 2017). This system classifies short Japanese text descriptions various items of family income and expenditures into more than 570 classes by applying the proposed classifier. The original classifier was developed in Perl in order to improve the efficiency for classifying short Japanese text descriptions.

The proposed classifier is modified from the original classifier so as to be easily applicable to short text descriptions with spaces between words in various languages, such as English. The proposed classifier is developed in R in order to explore its versatility and in consideration of the increasing number of R users in the field of official statistics. In addition, developing the system in R also has the following two advantages. First, R can be used with various operating systems, such as Windows, Linux, and OSX. Second, it is easy to implement programs and to move to the next process because R has good operability for data handling.

The remainder of this paper is organized as follows. The classical naïve Bayes classifier and the proposed classification method are presented in section 2. The experiments and results are described in section 3, and conclusions are presented in section 4.

2. CLASSIFICATION METHODS

The proposed classifier (Toko *et al*., 2017; Wada and Toko, 2017; Shimono *et al.*, 2017) is explained based on a method published by our previous study, Tsubaki *et al*. (2017). Their method is related with a simple naïve Bayes classifier for multiclass classification. In addition, they explained that their method is an extension of Taguchi's T method and standardized misclassification rate (Taguchi, 1997).

The naïve Bayes model (Spiegelhalter and Knill-Jones, 1984) has been commonly used in classification tasks for decades.

Let multinomial classes C take values in $\{1, ...,m\}$, and let $F = (F_1,..., F_J)$ be a J-dimensional random variable, the elements of which take values 0 or 1, which indicate the absence or presence, respectively, of a particular feature. If the j_{th} feature is present, then $f_j = 1$, otherwise $f_j = 0$. Under the naïve Bayes model, each feature is assumed to be conditionally independent of any other features given C. Thus, the conditional probability of the features of $\boldsymbol{F}$ given class C can be written as

$$P(F_1 = f_1, \dots, F_J = f_J | C = c) = \prod_{j=1}^{n} P(F_j = f_j | C = c) = \prod_{j=1}^{n} p_{cj}^{f_j} (1 - p_{cj})^{1-f_j}, \quad (1)$$

where $p_{cj} = p(F_j = 1 \,|C = c)$ for $c = 1, \dots, m$.

Then, let n_c be the number of instances in the training dataset in class c, and let n_{cj} be the number of instances in the training dataset in class c with $f_j = 1$. The maximum likelihood estimate (MLE) of p_{cj} can be written as

$$\hat{p}_{cj} = \frac{n_{cj}}{n_c}. \quad (2)$$

In order to prevent $\hat{p}_{cj}$ from being equal to 0 or 1, we added α to the denominator and β to the numerator:

$$\hat{p}_{cj} = \frac{n_{cj} + \beta}{n_c + \alpha}. \quad (3)$$

In the present study, we set α = 1 and β = 0.5. Then, under the assumption that $P(C = c) = p_c$, the posterior probability $P(C = c|\boldsymbol{F} = \boldsymbol{f})$ is proportional to

$$p_c \prod_{j=1}^{n} p_{cj}^{f_j} \left(1 - p_{cj}\right)^{1-f_j} , \quad (4)$$

and the naïve Bayes classifier classifies text into a class for which the features $\boldsymbol{f}$ give the highest posterior probability.

In Tsubaki *et al.* (2017), undersensitivities $\beta_{cj} = log\{p_{cj}/(1 - p_{cj})\}$ the posterior probabilities have been described as

$$P\left(C = c \middle| F_j = f_j, j = 1, \dots, J\right) = \frac{exp\left(\sum_{j=1}^{J} f_j \beta_{cj}\right)}{\sum_{k=1}^{m} exp\left(\sum_{j=1}^{J} f_j \beta_{kj}\right)} , \quad (5)$$

when prior probabilities $p_c \propto \{\prod_{j=1}^{n}\left(1 - p_{cj}\right)\}^{-1}$ for $c = 1, \dots, m.$

In the previous study (Toko *et al.*, 2017; Wada and Toko, 2017; Shimono *et al.*, 2017), we have simplified the process of assigning classes in the following concept. First, we considered as

$$\underset{c}{\text{argmax}}\, P\left(C = c \middle| F_j = f_j, j = 1, \dots, J\right) \propto \underset{c}{\text{argmax}} \prod_{\{j \,|f_j = 1\}} \frac{p_{cj}}{1 - p_{cj}}. \quad (6)$$

From (2), (3), and (6), it is seen that the maximum probability $P(C = c|\boldsymbol{F} = \boldsymbol{f})$ when f_j = 1 over m classes is influenced only by the amount of n_{cj}. Therefore, we have defined $\hat{p}_{cj}$ as follows (Toko *et al.*, 2017; Wada and Toko, 2017; Shimono *et al.*, 2017):

$$\hat{p}_{cj} = \frac{n_{cj} + \beta}{n_j + \alpha} , \qquad n_j = \sum_{c=1}^{m} n_{cj} . \quad (7)$$

In the present study, we set α = 1 and β = 0.5.

For comparison in terms of classification accuracy, we consider the classification and regression tree (CART) model and the random forest model.

The CART model, introduced by Breiman *et al.* (1984), is a commonly used algorithm in the field of data mining. The CART model predicts classification rules by using tree models from a training dataset. This model can take a set of discrete values and a set of continuous values as the target variable and the explanatory variable. A prediction tree with a discrete target variable is referred to as a classification tree, and a tree with continuous values is called a regression tree.

Random forests, proposed by Breiman (2001), are ensembles of tree predictors and are often used in classification, regression, and clustering tasks. Each tree depends on the values of a randomly selected explanatory variable

in randomly sampled instances of a training dataset. Random forests have the following advantages. Even if the data have numerous predictors, random forests can process training data in parallel because each tree is completely independent and uses Out-Of-Bag estimates to monitor error.

Since some packages for using these methods are available in CRAN, which is a network of ftp and web servers around world that store identical versions of code and documentation for R, in the present study we use rpart (ver. 4.1-11) and the randomForest (ver. 4.6-12) package for comparison with the proposed classifier.

3. EXPERIMENTS AND RESULTS

For the performance evaluation, we used the Stack Overflow dataset, a publicly available short description dataset that was published by Jiaming Xu *et al.* (2015) in Kaggle. The Stack Overflow dataset contains 20,000 instances, consisting of question titles in English and 20 different classification codes (see Table 1). In preparing the experimental dataset, we performed the following three tasks before processing. First, in order to achieve efficient feature extraction, we converted all letters in the question titles to lowercase. Second, we removed symbols that were clearly unnecessary for classification from the question titles. Finally, we randomly divided the dataset into 18,000 instances for training and 2,000 instances for evaluation. We repeated this division task three times and prepared three pairs of datasets for training and evaluation.

Overview of the Stack Overflow dataset

Table 1

No.	Class	Classification code	Number of instances in the dataset	No.	Class	Classification code	Number of instances in the dataset
1	wordpress	1	1,000	11	spring	11	1,000
2	oracle	2	1,000	12	hibernate	12	1,000
3	svn	3	1,000	13	scala	13	1,000
4	apache	4	1,000	14	sharepoint	14	1,000
5	excel	5	1,000	15	ajax	15	1,000
6	matlab	6	1,000	16	qt	16	1,000
7	visual-studio	7	1,000	17	drupal	17	1,000
8	cocoa	8	1,000	18	linq	18	1,000
9	osx	9	1,000	19	haskell	19	1,000
10	bash	10	1,000	20	magento	20	1,000

For another performance evaluation, the proposed classifier was applied to the Family Income and Expenditure Survey dataset. We randomly

extracted 11,000 instances of foodstuffs and dining-out data from the dataset. Then, we assigned 11 new labels to the dataset in order to evaluate the performance of the proposed classifier in larger-scale classification (see Table 2). We then randomly divided the dataset into 10,000 instances for training and 1,000 instances for evaluation. We repeated the above tasks three times and prepared three different datasets. Each instance of these datasets consists of an item name (a foodstuff name, a food product name, or an item on a restaurant menu) in Japanese and a label.

Overview of the Family Income and Expenditure Survey dataset

Table 2

No.	Class	Classification code	Number of instances in dataset 1	Number of instances in dataset 2	Number of instances in dataset 3
1	Cereals	A	1,018	1,007	1,049
2	Fish and shellfish	B	927	950	926
3	Meat	C	775	746	765
4	Dairy products and eggs	D	717	727	729
5	Vegetables and seaweed	E	2,966	2,954	2,913
6	Fruits	F	485	505	498
7	Oils, fats, and seasonings	G	661	713	686
8	Cakes and candies	H	1,026	1,025	1,048
9	Cooked food	I	1,221	1,211	1,270
10	Beverages, including alcoholic beverages	J	868	845	814
11	Meals outside the home	K	336	317	302

In the present study, the objective variable of classification is a short textual description. As described in section 2, we used the word-level N-gram model for feature extraction for the proposed classifier. For feature extraction, we performed the following two processes.

(1) Tokenization of each description by a morphological analyzer. We used MeCab (Kudo), which is a dictionary-attached morphological analyzer, to divide text descriptions into constituent words.

(2) For the proposed classifier, we took word-level N-grams ($N = 1,2,3 ...$) from the word sequences of text descriptions. In the present paper, we take 1-grams (any word) and 2-grams (any sequence of two consecutive words) as the features of an instance. We also take the whole-word sequence as a feature when the dataset is the Family Income and Expenditure Survey dataset. On the other hand, for the CART model and the random forest model, we simply take each constituent word as the feature of an instance because both models appear to connect each feature in the process of tree creation.

Each instance is assigned to the class that maximizes the probabilities calculated by means of the proposed method. We also fitted the random forest model and the CART model to the same data for comparison of the classification accuracy. Note that we applied the random forest model and the CART model without feature selection, in the same manner as for the proposed classifier. Table 3 shows the classification accuracy of each model. As shown by the experimental results obtained for the Stack Overflow dataset, the performance of the proposed classifier is better than the performance of the other two models. The performance of the proposed classifier for the Family Income and Expenditure Survey dataset is also better than the performance of the other two models in the case of the experiments using datasets 1 and 3, although the random forest method was slightly better than the proposed method in the case of the experiments using dataset 2.

Overview of classification accuracy

Table 3

		Stack Overflow	Family Income and Expenditure Survey
Proposed classifier	dataset 1	89.25%	84.20%
	dataset 2	89.30%	81.90%
	dataset 3	88.20%	83.90%
Random forest	dataset 1	86.10%	82.20%
	dataset 2	85.70%	82.20%
	dataset 3	86.00%	80.20%
CART	dataset 1	79.50%	51.30%
	dataset 2	79.60%	48.70%
	dataset 3	79.35%	46.80%

Tables 4 and 5 were compiled in order to compare these results in greater detail.

Table 4 lists the experimental results for the Stack Overflow dataset and shows the classification accuracy by classification codes. The classification accuracy was found to differ with the classification code. For example, the proposed classifier is effective at assigning correct codes for codes 1, 3, 5, 17, 18, and 20; however, it is not as efficient for codes 8 and 9. Moreover, the classification performances of both the random forest model and the CART model differ with the classification code. Although the performance of the proposed classifier is better than that of the random forest model, curiously the classes to which the proposed classifier is good at assigning correct codes

are roughly the same as those for the random forest model. In contrast, the classes for which the CART model is good at assigning correct codes differ from those with the proposed classifier and the random forest model, although the total classification accuracies of these models do not differ greatly.

Table 5 lists the experimental results for the Family Income and Expenditure Survey dataset and shows the classification accuracy by classification code. The tendency for the classification accuracy to differ by classification code is the same as the result with Stack Overflow data. Moreover, the proposed classifier and the random forest model have similar tendencies.

Stack Overflow: classification accuracy by classification code

Table 4

		Classification code									
		1	2	3	4	5	6	7	8	9	10
Proposed classifier	dataset 1	**95.1%**	91.4%	90.3%	85.6%	**96.7%**	84.5%	91.1%	72.6%	74.7%	78.4%
	dataset 2	93.7%	91.2%	**95.9%**	89.4%	94.9%	91.7%	80.2%	75.0%	75.2%	87.3%
	dataset 3	92.3%	84.6%	**96.6%**	85.6%	**95.4%**	**96.6%**	87.9%	61.2%	70.8%	81.9%
Random forest	dataset 1	93.2%	86.7%	91.3%	86.6%	**95.7%**	80.4%	87.1%	83.3%	69.5%	87.3%
	dataset 2	88.4%	86.3%	91.8%	88.5%	90.8%	87.0%	75.0%	86.0%	68.6%	90.2%
	dataset 3	92.3%	82.4%	91.0%	86.5%	91.7%	91.5%	85.9%	74.5%	70.8%	81.9%
CART	dataset 1	88.3%	83.8%	85.4%	76.3%	90.2%	76.3%	82.2%	**96.4%**	58.9%	69.6%
	dataset 2	87.4%	80.4%	92.9%	77.9%	81.6%	79.6%	68.8%	**96.0%**	61.0%	73.5%
	dataset 3	88.5%	78.0%	89.9%	73.1%	80.6%	87.3%	78.8%	91.8%	60.4%	68.1%

		Classification code									
		11	12	13	14	15	16	17	18	19	20
Proposed classifier	dataset 1	89.7%	90.6%	**95.0%**	90.2%	85.2%	92.6%	**95.1%**	**96.2%**	89.4%	**97.8%**
	dataset 2	93.1%	85.7%	93.6%	89.5%	86.1%	89.5%	93.3%	94.5%	91.8%	**95.7%**
	dataset 3	88.8%	89.3%	91.5%	92.7%	86.8%	90.2%	92.7%	92.7%	89.5%	**97.8%**
Random forest	dataset 1	81.3%	85.8%	88.0%	88.2%	78.7%	70.2%	91.2%	93.4%	84.6%	**95.7%**
	dataset 2	87.4%	86.8%	88.1%	85.7%	82.2%	73.7%	93.3%	90.1%	85.6%	91.4%
	dataset 3	81.3%	92.2%	85.1%	89.7%	83.3%	75.0%	90.6%	89.0%	86.3%	**96.7%**
CART	dataset 1	78.5%	79.2%	81.0%	75.5%	67.6%	62.8%	82.4%	89.6%	76.0%	92.4%
	dataset 2	77.0%	79.1%	81.7%	77.1%	75.2%	68.4%	86.5%	89.0%	76.3%	86.0%
	dataset 3	75.7%	84.5%	80.9%	75.3%	70.2%	69.6%	86.5%	83.5%	72.6%	91.3%

The Family Income and Expenditure Survey: classification accuracy by classification code

Table 5

		Classification code										
		A	B	C	D	E	F	G	H	I	J	K
Proposed classifier	dataset 1	83.7%	85.7%	**97.0%**	**96.8%**	**96.5%**	92.5%	83.9%	81.8%	71.4%	91.0%	84.0%
	dataset 2	90.7%	83.5%	**98.6%**	**97.1%**	93.6%	76.7%	71.9%	80.3%	68.3%	89.5%	55.0%
	dataset 3	86.2%	90.5%	**97.1%**	**98.1%**	**95.4%**	91.9%	77.3%	83.1%	67.0%	86.4%	71.4%
Random forest	dataset 1	80.4%	78.0%	91.5%	**95.2%**	**95.3%**	80.5%	74.6%	66.3%	56.4%	82.4%	70.4%
	dataset 2	86.4%	73.6%	**95.9%**	92.8%	**96.3%**	81.3%	67.2%	66.7%	63.7%	79.3%	55.0%
	dataset 3	79.6%	79.0%	92.9%	89.5%	**95.5%**	91.9%	64.2%	65.0%	53.9%	80.3%	75.9%
CART	dataset 1	51.1%	9.8%	49.3%	90.5%	**99.0%**	19.5%	0.0%	13.3%	9.6%	43.2%	44.4%
	dataset 2	51.9%	14.3%	46.6%	84.1%	**100.0%**	17.2%	0.0%	16.1%	13.7%	29.3%	35.0%
	dataset 3	49.0%	14.0%	44.3%	70.2%	**98.9%**	43.2%	3.0%	7.0%	17.4%	29.5%	31.0%

We also compared the proposed classifier to the other methods in terms of processing time and found that the proposed classifier can provide stable and quick processing. Table 6 shows the average over three runs of processing time (hh:mm:ss) of each model. In the experiments using the Stack Overflow dataset, although the processing time of the proposed classifier is longer than that of the CART model, the proposed classifier is much faster than the random forest model, even though the random forest uses 10 cores in processing. On the other hand, the experiments using the Family Income and Expenditure Survey, the proposed classifier is faster than the other methods.

Processing time (hh:mm:ss)

Table 6

	Setting	Stack Overflow	Family Income and Expenditure Survey
Proposed classifier	nonparallel	0:25:12	0:00:39
Random forest	10 cores, parallel processing	6:41:21	2:03:40
CART	nonparallel	0:11:40	0:02:43

4. CONCLUSION

We have presented a supervised multiclass classifier that employs the concept of the naïve Bayes model for the classification of short textual descriptions. As described in section 3, the proposed classifier works well regarding both classification accuracy and processing time. The proposed classifier could be applied to various coding tasks in practical situations.

For future research, we are considering publishing the proposed classifier on GitHub as an R-package in order to contribute to the handling of various coding tasks. However, before publication, we need to improve the operability of the proposed classifier as an R-package. In addition, the proposed classifier may need other improvement in order to improve usability

For example, using the proposed classifier in practical situations may require further improvement. First, the greater efficiency of each process of the proposed classifier must be improved in order to decrease the processing cost in terms of both memory usage and processing time. Although processing an R program is generally slower than processing programs developed in other languages, such as C and Java, the processing time can be shortened considerably by organizing the process efficiently. Such effort would also decrease the memory usage for processing, which depends on how the classifier is developed or organized. We should improve our classifier to process more efficiently because there would be strong demand for fast data processing with low memory usage by R.

Second, an additional function is needed for data pretreatment. We have recognized that it is necessary to realize an efficient classification process. However, thus far, we run minimum processes that convert English text descriptions into lowercase letters and remove clearly unnecessary symbols. Thus, we should consider "stop words", which are words to be filtered out before or after processing in the field of natural language processing. However, as described in section 1, we developed the proposed classifier for classifying short text descriptions with spaces between words in various languages, and it may be necessary to separate this additional function from the main classifier. Since proper data pretreatment depends on the dataset, the most proper data pretreatment must be applied for every type of dataset that is processed.

Finally, although the proposed system does not use the posterior probability of the most promising class as output data, the proposed classifier must be applied in practical situations, because the approximate accuracy of each assignment would be useful for finding a certain threshold for autocoding. This means that we can set a threshold by classification tasks for decision making whether or not an assigned class is automatically applied. In addition, it may be desirable to develop an additional function that suggests multiple labels when the posterior probability of the most promising class is lower than a certain threshold.

References

1. **Breiman, L.**, 2001, "Random Forest", Machine learning, 45, 1, 5–32.
2. **Breiman, L., Friedman, J., Stone, C.J.**, and **Olshen, R.A.**, 1984. "Classification and Regression Trees," Wadsworth, Belmont, CA, USA.
3. **Gweon, H., Schonlau, M., Kaczmirek, L., Blohm, M.**, and **Steiner, S**., 2017, "Three Methods for Occupation Coding Based on Statistical Learning." *Journal of Official Statistics*, 33, 1, 101-122. DOI: https://doi.org/10.1515/jos-2017-0006.
4. **Hacking, W. and Willenborg, L**., 2012. "Method Series Theme: Coding; interpreting short descriptions using a classification." Statistics Methods. Statistics Netherlands. Available at: https://www.cbs.nl/en-gb/our-services/methods/statistical-methods/throughput/throughput/coding (accessed September 2017).
5. **Kudo, T., MeCab**: Yet Another Part-of-Speech and Morphological Analyzer. Available at: http://taku910.github.io/mecab/ (accessed September 2017).
6. **Shimono, T., Toko, Y.,** and **Wada, K.,** 2017, "A supervised multiclass classifier equipped with trust-value calculation," submitted to *Journal of Official Statistics*.
7. **Spiegelhalter, D. J.** and **Knill-Jones, R. P.,** 1984, "Statistical and knowledge based approaches to clinical decision support systems, with an application in gastroenterology (with discussion)," Journal of Royal Statistical Society, Series A 147, 35-77.
8. T**aguchi, G.,** 1997, Mathematical for Quality Engineering – 7. Signal-to-Noise Ratio for Chemical and Biological Systems, Quality Engineering Forum, 5, 2, 3-9 in Japanese.
9. **Toko, Y., Shimono, T.,** and **Wada, K**. 2017, "Development and application of a simple machine learning algorithm for multiclass classifications," In Proceedings of New Techniques and Technologies for Statistics (NTTS2017), 13-17, Mar. 2017, Brussels, Belgium, Available at: https://ec.europa.eu/eurostat/cros/ntts2017programme/data/abstracts/abstract_61.html?zoom_highlight=machine (accessed September 2017).

10. **Tsubaki, H., Wada, K.**, and **Toko, T.** 2017, “An extension of Taguchi’s T method and standardized misclassification rate for supervised classification with only binary inputs,” to appear Proceedings of the ANQ Congress, 20-21, Sep. 2017, Kathmandu, Nepal.
11. **Wada, K**. and **Toko, Y.** 2017, “A supervised multiclass classifier for the family income and expenditure survey”, In Program and Book of Abstracts - Conference of the International Federation of Classification Societies IFCS-2017, pp. 285, 8-10, Aug. 2017, Tokyo, Japan.
12. **Xu, J.**, **Wang, P., Tian, G., Xu, B., Zhao, J., Wang, F.** and **Hao, H.** 2015, “Short Text Clustering via Convolutional Neural Networks.” In Proceedings of NAACL-HLT. May 31-June 5, 2015. 62-69. Denver, Colorado, USA.