International Journal of Fuzzy Logic and Intelligent Systems, vol. 8, no. 1, March 2008 pp. 52-60

Modified Version of SVM for Text Categorization

Taeho Jo

School of Computer and Information Engineering
Inha University
tjo018@naver.com

Abstract

This research proposes a new strategy where documents are encoded into string vectors for text categorization and modified versions of SVM to be adaptable to string vectors. Traditionally, when the traditional version of SVM is used for pattern classification, raw data should be encoded into numerical vectors. This encoding may be difficult, depending on a given application area of pattern classification. For example, in text categorization, encoding full texts given as raw data into numerical vectors leads to two main problems: huge dimensionality and sparse distribution. In this research, we encode full texts into string vectors, and apply the modified version of SVM adaptable to string vectors for text categorization.

Key Words : String Vector, Text Categorization, Support Vector Machine

1. Introduction

Text categorization refers to the process of assigning one or some of predefined categories to each document. Before doing the task, a fixed number of categories should be defined. The task is necessary for arranging documents based on their contents automatically for administrating textual information systems. Techniques of automatic text categorization have their high demand for both academic and industrial world for managing documents easily and efficiently. The scope of this research is restricted to automatic text categorization and electronic documents given as target for the task.

In 2002, Sebastiani stated that there are two classes of approaches to text categorization in his survey paper [10]. The first class of approaches is rule based ones given as heuristic ones. In this class of approaches, classification rules are defined manually in each category, in advance. This class of approaches was already applied to early text categorization systems [3]. However, this class of approaches requires prior knowledge to build classification rules; they have very good precision but poor recall with a lack of its flexibility.

Machine learning based approaches belong to the second class of ones to text categorization. In this class of approaches, classification rules or equations are defined automatically using sample labeled documents. In the previous class, classification rules are given manually as the input while in this class sample labeled documents are given as the input. This class of approaches has its better flexibility than rule based ones; it has slightly less precision but its much higher recall than rule based ones. Therefore, in recent text categorization systems, rule based approaches tend to be replaced by machine learning based ones [10].

It is required to represent documents into numerical vectors for using machine learning based approaches for text categorization. The representation leads to two main problems: huge dimensionality and sparse distribution. The dimension of numerical vectors representing documents is usually several hundreds, in spite of feature selection. When training examples are given as largely dimensional numerical vectors, it costs very much time for processing them, and a large number of training examples is required to build sufficient constraints proportionally to the dimension. An excessive reduction of dimension leads to the information loss by which classification performance is degraded very much.

The second problem in representing documents into numerical vectors is sparse distribution. It refers to the phenomena where each numerical vector has dominantly zero values as its elements. This problem indicates a poor discrimination among numerical vectors. It degrades classification performance very much. In order to mitigate this problem, a given text categorization is decomposed to binary classification problems in previous literatures [10] [11].

The idea of this research is to propose an alternative strategy of encoding documents, in order to address the two problems. In the proposed strategy, documents are encoded into string vectors, and a string vector refers to a finite ordered set of words. When in a numerical vector, numerical values given as its elements are replaced by words, it becomes a string vector. The goal of this research is to address the two problems by representing documents into string vectors, instead of numerical vectors. An

Manuscript received Nov. 21, 2007; revised Mar. 10, 2008.

additional advantage of string vectors is that they are more transparent than numerical vectors.

In this research, SVM (Support Vector Machine) is adopted as targets for modification into their adaptable versions to string vectors. In other words, we propose their modified versions where string vectors are used as their input vectors, instead of numerical vectors. A process of computing a semantic similarity between two string vectors is defined in this research as an operation on string vectors. Before performing the operation, we must build a similarity matrix from a corpus. Therefore, the operation on string vectors is performed, depending strongly on the similarity matrix.

This article consists of six sections including this section. In section 2, we will explore previous research on text categorization and an attempt to address the two problems. In section 3, we will describe two strategies of encoding documents for text categorization. In section 4, we will present architecture of text categorization systems and describe two versions of SVM. In section 5, we present experimental results of comparing the two versions of SVM with each other on two test beds and in section 6, mention the significance of this research and remaining tasks as the conclusion.

2. Previous Works

In this section, we will explore previous research on text categorization and a previous attempt to address the two problems in representing documents into numerical vectors. The scope of exploring approaches to text categorization is restricted to machine learning based ones, because they are more flexible than rule based ones. In this section, four supervised learning algorithms, NB (Naïve Bayes), KNN, SVM, and Back Propagation, are covered as main approaches to text categorization. In this section, each of them is described briefly and previous cases of applying it to text categorization are mentioned. We will justify why we select SVM as targets for the modification among the four approaches.

The fist typical approach to text categorization is KNN. KNN is a supervised learning algorithm where objects are classified based on target labels of their similar samples. The supervised learning algorithm has its two properties. Its first property is that it does not learn any training example until an unseen example is given; it is called lazy based learning algorithm [7]. Its second property is that it classified unseen objects based on target labels of their similar samples; it is called example based learning algorithm [7].

Among the four supervised learning algorithms, KNN was applied earliest to text categorization. In 1992, Massand et al applied it for classifying news articles [6]. The KNN has been fixed as the traditional machine learning based approach. In successive literatures, the KNN has been compared with other approaches [4] [10] [11] [12]. In 1999, Yang recommended the KNN as a good approach when she compared more than ten approaches [12].

Another popular approach to text categorization is NB. NB refers to a variant of Bayes classifier where each example is classified based on posterior probabilities of categories given it. In this approach, elements of an object are independent of each other and a probability of the object given a category is the product of probabilities of its elements given a category. The learning process of NB indicates is to define probabilities of all values of elements given categories using training examples. The NB learns training examples in advance, differently from the KNN.

Among the four approaches, NB has been most popularly applied for text categorization. In 1997, Mitchell mentioned NB is a typical approach to text categorization in his text book [7]. In 1999, Mladenic et al evaluated feature selection methods under the application of NB to text categorization [8]. In 2000, Androutsopoulos et al adopted NB as the approach to spam mail filtering [1]. Note that spam mail filtering is a practical and high demanding instance of text categorization.

The back propagation may be considered as another approach to text categorization. Among supervised neural networks, the back propagation is most popular model for classifications and regressions. It consists of three layers: the input layer, the hidden layer, and the output layer. In this neural network model, its weights are initialized randomly and output values are computed in a forward direction from the input layer to the output layer. The weights are updated to minimize errors between computed output values and target ones of training examples in a backward direction, from the output layer to the input layer.

In 1995, Winer applied the back propagation to text categorization in his master thesis [12]. He validated that the back propagation classifies unseen documents more accurately than KNN and NB on the standard test bed: Reuter 21578. In 2002, Ruiz et al proposed multiple back propagations in a hierarchical structure for text categorization [9]. They validated that the hierarchical structure of combining multiple back propagation is desirable than the flat one. However, the back propagation classifies documents more accurately, but it costs very much time for training it.

SVM is considered as a recent popular approach to text categorization. In the approach, unseen objects are classified by a linear combination of kernels of training examples. A kernel refers to a criterion of similarity between a training example and an unseen object. Depending on definition of kernels, SVM may be implemented as its various versions. Since SVM classifies unseen objects based on kernels of training examples, it is called kernel based learning [4].

In 1998, Joachim applied SVM to text categorization as a typical approach [5]. He proved that SVM is a suitable approach to text categorization by comparing it with NB and KNN. In 2000, Cristiani et al mentioned that SVM is a typical approach to text categorization in their text book. The reason that SVM becomes a recent popular approach to text categorization is that it is tolerable to huge dimension of numerical vectors which is the first problem from represent documents into numerical vectors.

In 2002, Lodhi et al attempted to solve the two problems in representing documents into numerical vectors by proposing the string kernel for SVM [5]. The string kernel proposed by them is the operation on full texts which computes a syntactic similarity between two full texts. Its additional advantage is that it is applicable independent of natural languages. Its disadvantage is that it takes too much time for perform the operation because of its very high complexity. Furthermore, their proposed version of SVM where a string kernel was used failed to be better than the traditional version of SVM.

SVM is adopted as the target for its modification in this research. The reason is that SVM is tolerable to huge dimension of numerical vectors and it becomes a popular approach, recently. SVM is modified into its adaptable version to string vectors by defining a kernel function of string vectors. The operation where a semantic similarity between two string vectors is computed is used as the kernel function in the modified version. As mentioned previously, depending on how to define a kernel function, SVM is implemented in its various versions.

3. Strategies of Encoding Documents

This section concerns two strategies of encoding documents for tasks of text mining, such as text categorization and text clustering. In the reality, documents given as raw data can not be processed directly by a computer. In this section, we will describe two strategies of encoding documents for processing them by a computer. One is the traditional strategy where documents are encoded into numerical vectors, and is described in section 3.1. The other is the proposed one where they are encoded into string vectors, and is described in section 3.2.

3.1. Traditional Strategy

This subsection concerns the traditional strategy of encoding documents for processing them. In this strategy, documents are encoded into numerical vectors for text categorization or text clustering. The process of doing that is described in detail in this subsection. For first, we will describe the process of extracting words as feature candidates from a corpus and the process of selecting some of them as features. For second, we will describe the process of assigning values corresponding to the features as final step of generating numerical vectors.

As the first stage of encoding documents into numerical vectors, feature candidates are extracted from a corpus. A collection of documents is given as a corpus in advance[1]. The corpus is given as input of this stage. A list of words and their frequencies is generated as its output. This stage consists of three steps, as illustrated in figure 1.

As illustrated in figure 1, a document or documents may be given as input of this stage. If more than two documents are given as the input, their full texts are concatenated into a full text. The integrated full text becomes the target for the tokenization. The full text is tokenized into tokens by a white space or a punctuation mark. Therefore, the output of this step is a list of tokens.

The next step to the concatenation & tokenization is the stemming & exception handling, as illustrated in figure 1. In this step, each token is converted into its root form. Before doing that, rules of stemming and exception handling are saved into a file. When the program encoding documents is activated, the rules are loaded into memory and the corresponding rules are applied to each token. The output of this step is a list of tokens converted into their root forms.

The last step of extracting feature candidates from a corpus is to remove stop words as illustrated in figure 1. Here, stop words are defined as words which function only grammatically without their relevance to content of their document; articles (a an, or the), prepositions (in, on, into, or at), pronoun (he, she, I, or me), and conjunctions (and, or, but, and so on) belong to this kind of words. It is necessary to remove this kind of words for more efficient processing. After removing stop words, frequencies of remaining words are counted. Therefore, a list of the remaining words and their frequencies is generated as the final output from the process illustrated in figure 1.

Since too many feature candidates are usually extracted from a corpus, some of them should be selected as features. Many schemes for selecting some of them were already proposed [8][10]. In this research, frequencies of words are set as the criteria for selecting features for simplicity. Words with their highest frequencies are selected as features[2]. Other schemes for selecting features will be used in future researches.

---

[1] In text categorization, sample labeled documents or separated collection of documents may be given as a corpus. Here, we set sample labeled documents as the corpus. However, unlabeled documents may be used as a corpus for extracting feature candidates.

[2] Stop words have their high frequencies in a given document or a collection of documents. However, since stop words were already removed in the process of extracting feature candidates, the kind of words never selected as features.

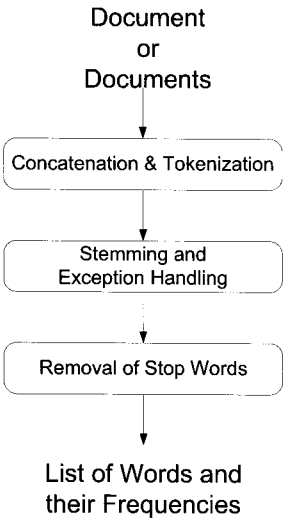

Figure 1. The Process of Extracting Feature Candidates

Once features are selected as attributes of numerical vectors, values should be assigned to the features. There are the three ways for assigning values to features for encoding documents into numerical vectors. For first, to each feature, a binary value is set, indicating its absence or presence; a document encoded into a binary vector in this way. For second, to each feature, its frequency in the document is set as its value; a numerical vector representing a document has integers as its elements, in this way. For third, we can set values of features as weights of words computed by equation (1),

$$weight_i(w_k) = tf_i(w_k)(\log_2 D - \log_2 df(w_k) + 1) \quad (1)$$

where $weight_i(w_k)$ indicates a weight of the word, w_k, which indicates its content based importance in the document, i, $tf_i(w_k)$ indicates the frequency of the word, w_k in the document, i, $df(w_k)$ is the number of documents including the word, w_k, and D is the total number of documents in a given corpus.

3.2. Proposed Strategy

This subsection concerns the proposed strategy of encoding documents. In this strategy, documents are encoded into string vectors, instead of numerical vectors. Depending on a given application area, it may be complicated or difficult that raw data are represented into numerical vectors for using machine learning algorithms. Especially in text mining, it is unnatural to encode documents into numerical vectors. The goal of this strategy is to address the two problems of the traditional strategy: huge dimensionality and sparse distribution.

A string vector is defined as a finite ordered set of words. If in a numerical vector, numerical values given as its elements are replaced by words, it becomes a string vector. A d-dimensional string vector is notated by $[w_1, w_2, ..., w_d]$. Although two string vectors have their identical elements, if their elements are in different order, they are different string vectors from each other. Therefore, a string vector should be distinguished from a bag of words based on this fact.

Properties of words may be set as features of string vectors. Features of string vectors are defined in one or combined one of three views. In the first views, features are defined based on posting information of words: a random word in the first sentence, a random word in the last sentence, and a random word in the first paragraph. In the second view, they are defined based on linguistic properties of words, such as first noun, first verb, last noun, and last verb. In the third view, they are defined based on their frequencies, such as the most frequent word, the second most frequent word, and the third most frequent word, and so on.

In this research, the third way of defining features of string vectors is adopted; a strong vector consists of words in the descending order of their frequencies. The reason of defining features of string vectors so is to implement the encoder easily and simply. Figure 2 illustrates the process of encoding documents into string vectors. A document is given as the input. The process generates a string vector as its output.

The process of encoding a document into a string vector consists of the three steps, as illustrated in figure 2. The first step, indexing, was already explained in detail in section 3.1 and illustrated in figure 1. In the second step, the most frequent words are selected as elements with their fixed number; the number indicates the dimension of string vectors given as a parameter. The selected words are sorted in the descending order of their frequencies and they are generated as a string vector.

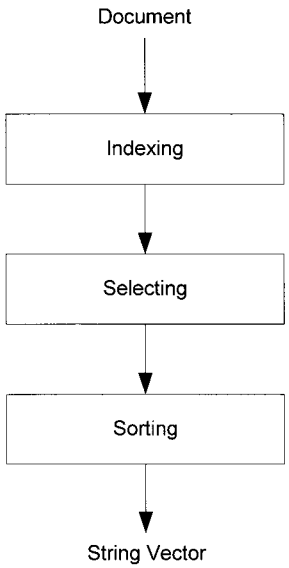

Figure 2. The Process of Encoding Documents into String Vectors

In order to define an operation on string vectors for the modified version of SVM, a similarity matrix should be built from a corpus as follows.

$$\begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1N} \\ s_{21} & s_{22} & \cdots & s_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ s_{N1} & s_{N2} & \cdots & s_{NN} \end{bmatrix}$$

A similarity matrix indicates semantic similarities of all possible pairs of words included in a corpus as a squire matrix. Each element in the matrix means a semantic similarity between two words corresponding to a column and a row. The semantic similarity is computed by equation (2),

$$s_{ij} = ss(w_i, w_j) = \frac{2df(w_i, w_j)}{df(w_i) + df(w_j)} \quad (2)$$

where s_{ij} is a semantic similarity between the two words, w_i and w_j, $df(w_i)$ is a number of documents including the word in the corpus, w_i, and $df(w_i, w_j)$ is a number of documents including both words, w_i and w_j. Equation (2) implies that a semantic similarity between two words is determined based on their collocations within a document.

The similarity matrix defined as a basis for the operation on string vectors has two properties. The first property is that the similarity matrix has 1.0 in its diagonal elements.

Property 1. $s_{ii} = 1.0 \ (1 \le i \le N)$

Since two words at position, i are a word and itself, the column and the row are identical to each other and the similarity as computed with equation (2) is 1.0. Therefore the similarity matrix's its diagonal elements are 1.0.

<Proof>

$$ss(w_i, w_i) = \frac{2df(w_i, w_i)}{df(w_i) + df(w_i)} = \frac{2df(w_i)}{2df(w_i)} = 1.0$$

The second property is that the similarity matrix is a symmetry matrix by commutative law in equation (2).

Property 2. $s_{ij} = s_{ji} (1 \le i, j \le N)$

Since the operation of computing the similarity of two words, w_i and w_j using equation (2) is commutative, the similarity matrix is symmetric.

<Proof>

$$s_{ij} = ss(w_i, w_j) = \frac{2df(w_i, w_j)}{df(w_i) + df(w_j)} = \frac{2df(w_j, w_i)}{df(w_j) + df(w_i)} = ss(w_j, w_i) = s_{ji}$$

The operation on string vectors involved in the modified version of SVM is defined based on the similarity matrix. The operation is the process of computing a semantic similarity between two string vectors. The operation is defined by equation (3),

$$\boldsymbol{s}_i = [w_{i1}, w_{i2}, \ldots, w_{id}], \boldsymbol{s}_j = [w_{j1}, w_{j2}, \ldots, w_{jd}]$$
$$sim(\boldsymbol{s}_i, \boldsymbol{s}_j) = \frac{1}{d} \sum_{k=1}^{d} ss(w_{ik}, w_{jk}) \quad (3).$$

In the proposed version of SVM, it is used as a kernel function of string vectors.

4. Text Categorization Systems

This section concerns architecture of text categorization systems and the modified version of SVM. The two approaches involve trainer and classifier as the engine in the text categorization system. The reason of adopting SVM is that they are modified into their adaptable versions to string vectors easily and simply. If the operation is used as a kernel function on string vector, SVM can be modified so.

Figure 3 illustrates the architecture of text categorization systems consisting of encoder, trainer, and classifiers as their modules. The encoder given as the interface to input data maps documents into numerical vectors or string vectors; the strategies of implementing it were described in detail in section 3. The trainer builds classification capacity using training documents and provides it for the classifier. The classifier classifies unseen documents using the classification capacity given as classification rules or equations. SVM is used as a scheme for implementing the two modules in this research and both of them are described in two subsection.

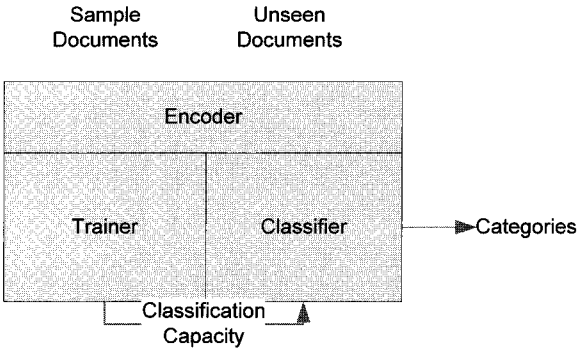

Figure 3. Architecture of Text Categorization Systems

4.1. Support Vector Machine

This subsection concerns a brief description of two versions of SVM. Recently, SVM becomes a very popular approach to classification problems including text categorization. SVM is applied to not only text categorization but also any other application areas such as protein classifications and image classifications. SVM is a supervised learning algorithm where objects are classified by a linear combination of kernels of training examples. SVM is regarded as a theoretically and

practically sound classification algorithm [2].

In the primitive version of SVM, an unseen example, $\mathbf{x}$ is classified by a linear combination of its inner products with training examples, as expressed in equation (6),

$$f(\boldsymbol{x})=\begin{cases}-1 & \sum_{i=1}^{m}\alpha_i(\boldsymbol{x}\cdot\boldsymbol{x}_i)+b<0\\ 1 & \sum_{i=1}^{m}\alpha_i(\boldsymbol{x}\cdot\boldsymbol{x}_i)+b\geq 0\end{cases} \quad (4)$$

where x_i indicates a training example. In equation (6), α_i corresponding to the training example, x_i, is a Lagrange multiplier. The learning process of SVM is to optimize the Lagrange multipliers to reduce error and maximize the margin, in classifying training examples. The function expressed in equation (6) generates -1 indicating the negative class and 1 indicating the positive class; SVM is used as a binary classifier. Optimized Lagrange multipliers are given as non-zeros or zeros; training examples with their non-zero corresponding Lagrange multipliers are called support vectors [2].

If a distribution of training examples is not linearly separable, they are mapped into another space where their distribution is linearly separable. In the general version of SVM, the inner product of two mapped examples, $F(x)$ and $F(x_i)$, is computed. Note that an inner product of two vectors in any space is absolutely given as a scalar value. As expressed in equation (7), the inner product of two mapped vectors is computed without mapping them explicitly using a kernel function.

$$F(\boldsymbol{x})\cdot F(\boldsymbol{x}_i)=K(\boldsymbol{x},\boldsymbol{x}_i) \quad (5)$$

Therefore, in the general version of SVM, the classification function is given as equation (8).

$$f(\boldsymbol{x})=\begin{cases}-1 & \sum_{i=1}^{m}\alpha_i K(\boldsymbol{x}\cdot\boldsymbol{x}_i)+b<0\\ 1 & \sum_{i=1}^{m}\alpha_i K(\boldsymbol{x}\cdot\boldsymbol{x}_i)+b\geq 0\end{cases} \quad (6)$$

The trainer implemented with SVM learns encoded sample documents in advance. Its learning process is the optimization of Lagrange multipliers, as mentioned above. In this research, SMO (Sequential Minimal Optimization) algorithm is adopted as the approach to optimization, because of its popularity. It is skipped in this article, and it is explained in detail in the literature [4]. The trainer generates the optimized Lagrange multipliers as its output.

The classifier implemented with SVM takes the optimized Lagrange multipliers and encoded training examples from the trainer as its input. Note that a SVM is for a binary classification; if the SVM is applied to text categorization, the task should be decomposed into binary classification tasks as many as predefined categories. Each SVM given as a binary classifier is assigned to each category; it determines whether an encoded unseen document belongs to the corresponding category or not. Training documents are labeled with the positive class indicating 'belonging' or the negative class indicating 'not belonging'. In this research, text categorization system implemented with SVM is applied category by category.

The proposed version of SVM uses a kernel function of string vectors, not of numerical vectors. The operation expressed in equation (3) is used as a kernel function for the proposed version of SVM. The kernel function of string vectors should be distinguished from the string kernel proposed by Lodhi et al in 2002. In the Lodhi et al's work, the string kernel is the kernel function of two full texts given as raw data. In this research, the version of SVM is modified into the adaptable one to string vectors by using a kernel function of them.

Note that SVM is a kernel based learning algorithm [4]. In this class of machine learning algorithms, a kernel refers to a boundary of influences of training examples on classification of unseen objects. Kernel based learning is defined as the paradigm of machine learning where classification rules or equations are defined based on the boundaries. In both versions of SVM, an inner product of two numerical vectors and a semantic similarity of two string vectors are given as kernels. Once SVM is modified so, it is expected to modify other kernel based learning algorithms.

5. Experiment Results

This section concerns experiments where two strategies of encoding documents for text categorization. Two collections of news articles, NewsPage.com and Reuter21578, were used as the test beds. In these experiments, documents are encoded into numerical vectors with high dimensions in the traditional strategy, while they are encoded into string vectors with low dimensions in the proposed one. We used SVM to text categorization. The goal of these experiments is to observe whether the proposed versions of both machine learning algorithms are better than or comparable to their traditional ones with smaller sizes of inputs.

5.1. NewsPage.com

This set of experiments concerns comparisons of two strategies of encoding documents on the first test bed, NewsPage.com. The test bed, NewsPage.com, consists of five categories and 1,200 news articles. We use SVM as a typical approache to text categorization in this set of experiments, as mentioned in section 2. The similarity matrix is built from a collection of training documents given as a corpus, in order to perform the operation on string vectors. The significance of this

set of experiments is the first step of testing which of the two strategies of encoding documents is better.

The first test set for evaluating the two strategies of encoding documents is NewsPage.com. This test bed consists of 1,200 news articles in plain ASCII text files built by copying and pasting news articles in the web site, www.newspage.com. Table 1 shows the predefined categories, the number of documents of each category, and the partition of the test bed into training set and test set. As shown in table 1, the ratio of training set to test set is set as 7:3. Here, the test bed was named after the web site.

Table 1. Training Set and Test Set of Newspage.com

Category Name	Training Set	Test Set	#Document
Business	280	120	400
Health	140	60	200
Law	70	30	100
Internet	210	90	300
Sports	140	60	200
Total	840	360	1200

Table 2 illustrates configurations of parameters of the two approaches, dimensions, and similarity matrix for this set of experiments. From a given training set, words with highest frequency are selected as features indicating attributes of numerical vectors. In the traditional strategy, documents are encoded into numerical vectors with 100, 250, and 500 dimensions as large input sizes, and in the proposed strategy, they are encoded into string vectors with 10, 25, and 50 dimensions, as small input sizes. We set the size of similarity matrix to 350 X 350, as illustrated in table 2, because of constraints of time and system resources.

Table 2. Configurations for this Set of Experiments

Support Vector Machine	#iterations = 200 |C| = 4
Dimensions	Numerical Vectors: 100, 250, and 500 String Vectors: 10, 25, and 50
Size of Similarity Matrix	350 X 350

In figure 4 and 5, the results of comparing the two strategies of encoding documents with each other are presented into bar graphs. In x-axis of each graph, each group indicates a strategy of encoding documents, and an individual graph within a group indicates a dimension of numerical vectors or string vectors. A given text categorization on this test bed is decomposed into five binary classifications, and for each category, F1-measure into which recall and precision are combined with each other is used as the evaluation measure. Therefore, y-axis indicates micro-averaged F1 in the left side and macro-averaged F1 in the right side. Micro-averaged F1 is obtained by computing F1 measure based on the total number of true positives, classified positives, and correctly classified positives over the five categories, while macro-averaged F1 is obtained by averaging five F1 measures of the five categories.

Figure 4 illustrates the results of comparing the two strategies of encoding documents in using SVM for text categorization on this test bed. In the traditional strategy, both measures are almost same to each other over all dimensions of numerical vectors. In the proposed strategy, when documents are encoded into 50 dimensional string vectors, both measures are highest. In both measures, when documents are encoded into 10 and 25 dimensional string vectors, the proposed strategy is comparable to the traditional one. In both measures, when documents are encoded into 50 dimensional string vectors, the proposed strategy is better than the traditional one by 35%, as illustrated in figure 5.

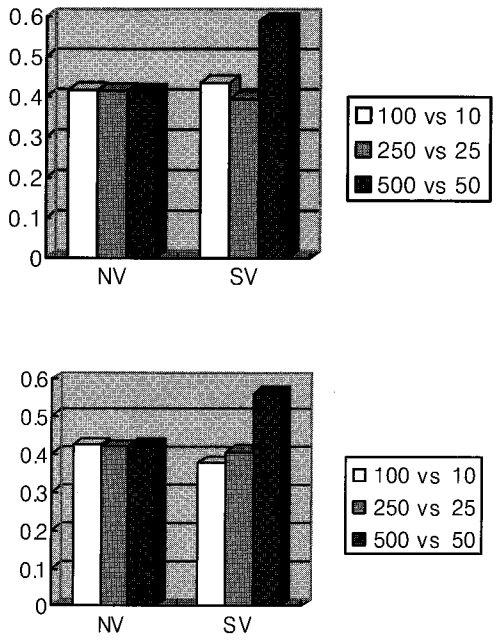

Figure 4. The Results of Two Strategies of Encoding Documents in using SVM on NewsPage.com
Microaveraged-F1 Measure (Left) and Macroaveraged-F1 Measure (Right)

5.2. Reuter21578

This section concerns another set of experiments where two strategies of encoding documents are compared with each other on another test bed. The test bed used in this set of experiments is Reuter21578 which has been used very popularly as the standard test bed for evaluating approaches to text categorization [10]. Dimensions of numerical vectors and string vectors are set identically to those in the previous set of

experiments. SVM is used for this set of experiments with the identical configurations in the previous set. The goal of this set of experiments is to evaluate the two strategies of encoding documents one more time for more confirmative judgments.

The test bed used in this set of experiments is a collection of news articles called Retuer21578. Although this test bed contains more than 100 categories, we selected only ten most frequent categories for running this set of experiments3. The text categorization on this test bed is decomposed into ten binary classifications inconsistently with the number of predefined categories. Table 3 illustrates the partition of this test bed into training and test set. The size of training set is different to each category, as illustrated in table 3.

Table 3. Partition of Training Set and Test Set in Reuter21578

Category Name	Training Set	Test Set	#Document
Acq	1452	672	2124
Corn	152	57	209
Crude	328	203	531
Earn	2536	954	3490
Grain	361	162	523
Interest	296	135	431
Money-Fx	553	246	799
Ship	176	87	263
Trade	335	160	495
Wheat	173	76	249

Figure 5 illustrates the results of comparing the two strategies of encoding documents in using SVM for text categorization. In the traditional strategy, when documents are encoded into 100 dimensional numerical vectors, micro-averaged F1 is highest, and when documents are encoded into 250 dimensional vectors, macro-averaged F1 is highest. In the proposed strategy, when documents are encoded into 50 dimensional string vectors, both micro-averaged F1 and macro-averaged F1 are highest. As illustrated in figure 4, when SVM is used as an approach to text categorization, the proposed strategy of encoding documents is better than the traditional strategy. In using SVM, the proposed strategy satisfies our expectation sufficiently.

5.3. Final Discussion

Let's consider why the proposed strategy was effective only in using SVM for text categorization. The cause of the effectiveness is that string vectors are free from sparse distributions which happens frequently in numerical vectors representing documents. SVM uses the operation on numerical vectors, inner product between two numerical vectors. If numerical vectors given as training and test examples are sparse, the operation generates zero values very frequently. The fact that the sparse distribution of numerical vectors degrades the performance of text categorization came true in using SVM under the traditional strategy of encoding documents.

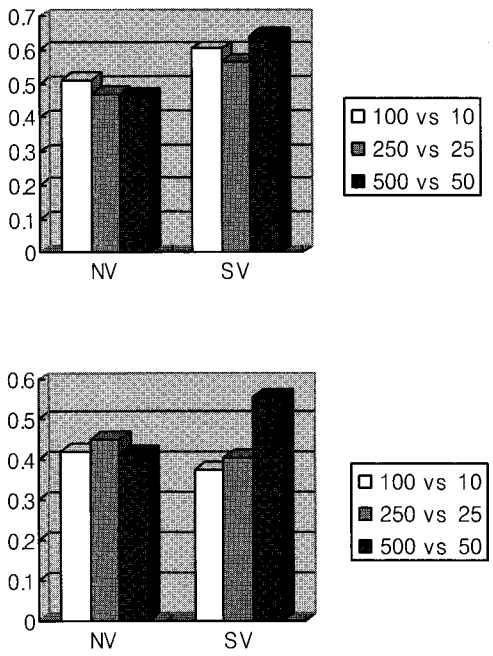

Figure 5. The Results of Two Strategies of Encoding Documents in using SVM on Reuter21578
Microaveraged-F1 Measure (Left) and Macroaveraged-F1 Measure (Right)

According to the results in these experiments, we can judge that the information loss caused by using the small sized similarity matrix should be addressed, in order to improve the proposed strategy. The solution is to use inverted indices of words for performing operations on string vectors, instead of a similarity matrix. We can compute a semantic similarity between two string vectors directly from the inverted indices, without building any similarity matrix. Note that there is the trade-off between a similarity matrix and the inverted indices. If the inverted indices are used for performing operations on string vectors, the reliability is expected to be improved, but it costs more time for performing the operations.

6. Conclusion

In this research, we proposed an alternative strategy of encoding documents, in order to address the two problems from

[3] In the rest, each category has a very sparse number of news articles.

the traditional strategy: huge dimensionality and sparse distribution. The two machine learning algorithms, SVM, were modified into their adaptable versions to string vectors. The process of computing a semantic similarity between two string vectors is defined as an operation on string vectors. In the modified version of SVM, it is used as a kernel function of two string vectors.

Once a similarity matrix is built from a corpus, it may be used continually while the domain of documents is not changed. However, note that it costs very much time and system resources for building it. Therefore, as mentioned in section 5, the size of the similarity matrix is restricted to 350 by 350. The restriction leaded to a great information loss. This is the reason that the proposed versions were not better than the traditional versions, as presented in section 5.

Other machine learning algorithms such as Naïve Bayes and back propagation may be considered to be modified into their adaptable versions to string vectors. SVM is modified easily once the process of computing a semantic similarity between two vectors is defined as the operation. The operation may be insufficient for modifying other machine learning algorithms. For example, it requires the definition of a string vector which is representative of string vectors corresponding to a mean vector in numerical vectors for modifying a k-means algorithm into the adaptable version. Various operations on string vectors should be defined in a future research for modifying other machine learning algorithms.

7. Literatures

[1] Androutsopoulos, K. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos, "An Experimental Comparison of Naïve Bayes and Keyword-based Anti-spam Filtering with personal email message", *The Proceedings of 23rd ACM SIGIR*, pp160-167, 2000.

[2] M. Hearst, "Support Vector Machines", *IEEE Intelligent Systems*, Vol 13, No 4, pp18-28, 1998.

[3] P. Jackson, and I. Mouliner, *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization,* John Benjamins Publishing Company, 2002.

[4] T. Joachims, "Text Categorization with Support Vector Machines: Learning with many Relevant Features", *The Proceedings of 10th European Conference on Machine Learning*, pp143-151, 1998.

[5] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, Text Classification with String Kernels, *Journal of Machine Learning Research*, Vol 2, No 2, pp419-444, 2002.

[6] B. Massand, G. Linoff, and D. Waltz, "Classifying News Stories using Memory based Reasoning", *The Proceedings of 15th ACM International Conference on Research and Development in Information Retrieval*, 1992, pp59-65, 1992.

[7] Mitchell, T. M., *Machine Learning*, McGraw-Hill, 1997.

[8] D. Mladenic, and M. Grobelink, "Feature Selection for unbalanced class distribution and Naïve Bayes", *The Proceedings of International Conference on Machine Learning*, pp256-267, 1999.

[9] M. E. Ruiz, and P. Srinivasan, "Hierarchical Text Categorization Using Neural Networks", *Information Retrieval*, Vol 5, No 1, 2002, pp87-118, 2002.

[10] F. Sebastiani, "Machine Learning in Automated Text Categorization", *ACM Computing Survey*, Vol 34, No 1, pp1-47, 2002.

[11] E. D. Wiener, "A Neural Network Approach to Topic Spotting in Text", *The Thesis of Master of University of Colorado*, 1995.

[12] Y. Yang, "An evaluation of statistical approaches to text categorization", *Information Retrieval*, Vol 1, No 1-2, pp67-88, 1999.

Taeho Jo

Taeho Jo received PhD degree from University of Ottawa in 2006. Currently, he works for Inha University as a professor. He has submitted and published more than 100 research papers to journals and proceedings since 1996. Previously he has ever worked for industrial organizations: Samsung, ETRI, KISTI, and KAIST Institute for IT Convergence. His research interests are text mining, neural networks, machine learning, and information retrieval.