DOI : 10.3745/JIPS.2008.4.1.017

Journal of Information Processing Systems, Vol.4, No.1, March 2008 **17**

Inverted Index based Modified Version of KNN for Text Categorization

Taeho Jo*

Abstract: This research proposes a new strategy where documents are encoded into string vectors and modified version of KNN to be adaptable to string vectors for text categorization. Traditionally, when KNN are 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 modify the supervised learning algorithms adaptable to string vectors for text categorization.

Keywords: *String Vector, K- Nearest Neighbor, Text Categorization*

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 [16]. 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 [6]. 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 [16].

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 [16][17].

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 additional advantage of string vectors is that they are more

Manuscript received November 29, 2007; revised February 13, 2008; accepted March 10, 2008.

Corresponding Author: Taeho Jo

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

transparent than numerical vectors.

In this research, KNN (K Nearest Neighbor) 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.

However, there was a previous attempt to use the modified versions of SVM and KNN for text categorization. In the previous attempt, a restricted similarity matrix was used as a basis for the operation on string vectors. Once a similarity matrix is built from a corpus, it is easy and fast to perform the operation. However, it cost very much in terms of time and system resources to build the similarity matrix; if the numbers of words and documents in a corpus is $_N$ and M, respectively, the complexity for doing that becomes $O(M^2N^2)$. If more than 10,000 words and 1,000 documents are given, it is almost impossible to build a full similarity matrix in our reality.

In this attempt, we will use an inverted index as the basis for the operation on string vectors involved in the modified version. An inverted index refers to a list of words each of which is linked to a list of documents including it. The advantage of an inverted index over a similarity matrix is that it is cheaper to build an inverted index from a corpus than a similarity matrix. The complexity of doing that reduces to $O(MN)$. Therefore, since it is possible to build a full inverted index from a corpus, in this research, it is expected to avoid the information loss from using a restricted sized 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 KNN. In section 5, we present experimental results of comparing the two versions of KNN with each other on two test beds and in section 6, mention the significance of this research and remaining tasks as the conclusion.

2. Related Work

In this section, we will explore previous research on text categorization and previous attempts 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, the four supervised learning algorithms, NB (Naïve Bayes), KNN, SVM, and Back Propagation, are covered as the popular and representative approaches to text categorization. In this section, we will describe briefly each of them and mention previous cases of applying it to text categorization. We will justify in detail why we select KNN and 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 by voting target labels of their most 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 [13]. Its second property is that it classified unseen objects based on target labels of their similar samples; it is called example based learning algorithm [13].

Among the four supervised learning algorithms, KNN was applied earliest to text categorization. In 1992, Massand et al applied it for classifying news articles [12]. Since then, the KNN has been assigned as the traditional machine learning based approach. In successive literatures, the KNN has been compared with other approaches, and it has been insisted that their own approaches are better than KNN in text categorization [7] [16] [17] [18]. However, in 1999, Yang recommended the KNN as a good approach when she compared more than ten approaches [18].

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 assumed to be independent of each other, and a probability of the object given a category is given as the product of probabilities of its elements given a category. The learning process of NB 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 [13]. In 1999, Mladenic et al evaluated feature selection methods, implying that the NB is the most popular approach to text categorization [14]. In 2000, Androutsopoulos et al adopted NB as the approach to spam mail filtering [1]. Note that spam mail filtering is a practical and highly 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 attempted to apply the back propagation to text categorization in his master thesis [17]. 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 the fashion of a hierarchical structure for text categorization [15]. They validated that the hierarchical combination of multiple back propagations is desirable than the flat one [15]. However, note that it costs very much time for training the back propagation.

Recently, the SVM becomes a popular approach to any classification problem including text categorization. In the SVM, unseen objects are classified by a linear combination of kernels of training examples. A kernel function in the SVM indicates a criterion of similarity between a training example and an unseen object. Depending on how to define a kernel function, SVM may be implemented as its various versions. Since the SVM classifies unseen objects based on a kernel function of training examples, it is called a kernel based learning algorithm [2].

In 1998, Joachim attempted initially to apply the SVM to text categorization [7]. He validated empirically that the SVM is a suitable approach to text categorization by comparing it with NB and KNN. In 2000, Cristiani et al mentioned that the SVM is a typical approach to text categorization in their text book. In 2002, Ducker et al used SVM for spam mail filtering as a practical instance of text categorization and compared it with NB [3]. The reason that the SVM becomes more accurate to text categorization than any other approach is that it is tolerable to huge dimension of numerical vectors; it addresses the first problem: huge dimensionality.

In 2002, Lodhi et al attempted to solve the two main problems from representing documents into numerical vectors by proposing the string kernel for SVM [11]. The string kernel proposed by them is the operation on full texts where a syntactic similarity between two full texts is computed. Its additional advantage is that it is applicable independent of natural languages without considering their grammatical properties. Its disadvantage is that it takes too much time for performing the operation because of its very high complexity. Furthermore, their proposed version of SVM where the string kernel was used failed to be better than the traditional version of SVM.

In 2005, NTSO (Neural Text Self Organizer) was proposed as a solution to the two problems by Jo and Japkowicz. NTSO is an unsupervised neural network which follows learning rule of Kohonen Networks and uses string vectors as its input vectors. Two operations are involved in training the neural network. The first operation is the process of computing a semantic similarity between two string vectors; it is also used in the proposed versions of SVM and KNN in this research. The second operation is the process of retrieving a set of inter-words between two words which are words with higher semantic similarities than that between the two words; the operation has very high complexity in NTSO.

With two reasons, KNN is adopted as target of its modification in this research. The first reason is that KNN is simple and easy to modify it into its adaptable version to string vectors. Once a similarity between two string vectors is able to be defined, KNN is modified easily. The second reason is that KNN was recommended previously as a practical approach. In 1999, Yang considered KNN as one of recommended approaches among more than ten approaches to text categorization [18], and in 2002, Sebatiani recommended it, since it is simple and comparable to the best approach, SVM, on the standard test bed, Reuter 21578 [17].

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 our reality, it is impossible that documents given as raw data are processed directly by a computer. In this section, we will describe the two strategies of encoding documents to be enabled to process them by a computer. One is the traditional strategy where documents are encoded into numerical vectors, and it is described in section 3.1. The other is the proposed one where they are encoded into string vectors, and it is described in section 3.2.

3.1. Numerical Vectors

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. In this section, we will describe in detail the process of doing that. 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 the final step of generating numerical vectors.

In 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[3]. A particular corpus is given as the 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; here, documents are given as the input, since they are given as a corpus. The full texts of the documents are concatenated into a full text and it is the target for the tokenization. The concatenated full text is tokenized into tokens by a white space or a punctuation mark. Therefore, the output of the first step of this stage is a list of tokens.

The next step is the stemming & exception handling, as illustrated in Figure 1. In this step, each token is converted into its root form. In advance, rules of stemming and exception handling are saved into a file. When the program which encodes documents is executed, the rules are loaded into memory and the corresponding rules are applied to each token. The output of this step is a list of root forms of tokens.

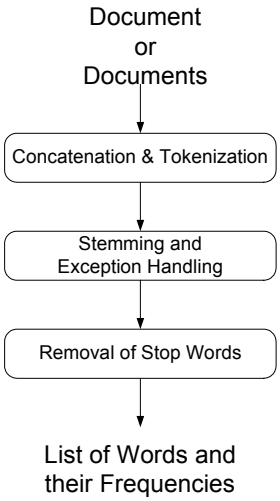

Fig. 1. The Process of Extracting Feature Candidates

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 perform only grammatical functions without their relevance to content of a document or documents; 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 stage illustrated in Figure 1.

Since too many feature candidates are usually extracted from a corpus, some of them should be selected as features[4]. Many schemes for selecting some of them were already proposed [14][16]. In this research, for a simple implementation, features are selected by their frequencies. In other words, words with their highest frequencies are selected as features[5]. Other schemes for selecting features will be used in our future researches.

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 in encoding documents into numerical vectors. For first, to each feature, a binary value which indicates whether its corresponding feature is absent or present in the document as the source; 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),

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

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. String Vectors

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

[1] In text categorization, the training collection or a separated collection may be given as a corpus. Here, we set the former as the corpus. However, note that it is possible to use unlabeled documents as a corpus for extracting feature candidates.

[2] Usually, more than 10,000 words are extracted as feature candidates. The number of selected features becomes usually several hundreds.

[3] 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.

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 numerical values given as its elements in a numerical vector are replaced by words, the numerical vector becomes a string vector. A d-dimensional string vector is notated by $[w_1, w_2, ..., w_d]$. For example, [computer system information] is an instance of a three dimensional string vector. Note that the string vector, [computer system information] is different from the string vector [system computer information], since elements are dependent on their positions like the case in every numerical vector.

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 easily and simply the encoder of a text clustering system. Figure 2 illustrates the process of encoding documents into string vectors. A document is given as the input. The process illustrated in figure 2 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.

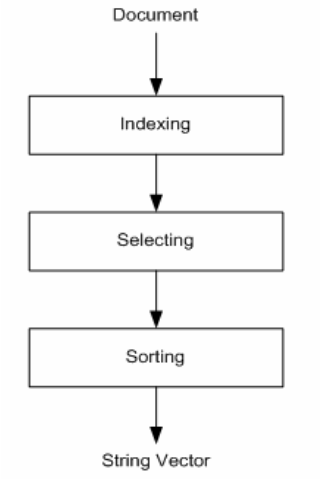

Fig. 2. The Process of Encoding Documents into String Vectors

As mentioned in section 1, an inverted index is used as the basis for the operation on string vectors as expressed in equation (3). An inverted index is defined as a list of words each of which is linked with a list of documents including it. Figure 3 illustrates the data structure of an inverted index. As illustrated in figure 3, each word is linked with a list of document identifiers including the word. A list of words is implemented with a hash table, while a list of documents which including a word is implemented with an array.

A semantic similarity between two words is computed based on a number of documents where both words are collocated with each other. The more documents including both words, the higher semantic similarity between them is. From the inverted index, two lists of document identifiers corresponding to the two words are retrieved. The intersection is taken from the two lists of document identifiers as a list of documents including both words. Therefore, 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.

The operation on string vectors involved in the modified version of KNN and SVM is defined based on an inverted index illustrated as an example in figure 3. The operation is the process of computing a semantic similarity between two string vectors. The operation is defined by equation (3),

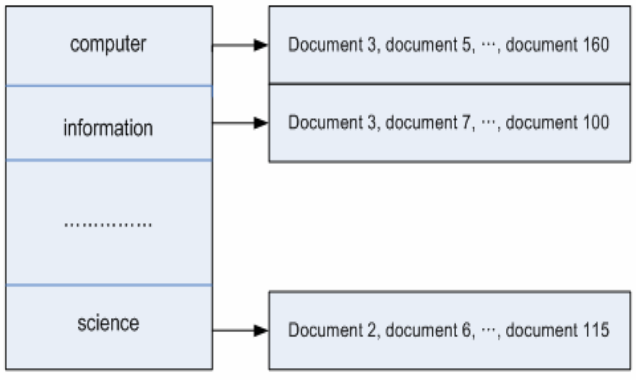

Fig. 3. Inverted Index

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

In the proposed version of KNN, this operation is used as a similarity measure between a training example and an unseen example. In the proposed version of SVM, the operation expressed in equation (3) is used as a kernel function of string vectors.

4. Text Categorization Systems

This section concerns architecture of text categorization systems KNN. The approach involves trainer and classifier as the engine in the text categorization system. The reason of adopting KNN is that they are modified into their adaptable versions to string vectors easily and simply. If the defined operation on string vectors is used as similarity measure between a training document and an unseen document encoded into string vectors, KNN 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. KNN is used as a scheme for implementing the two modules in this research and both of them are described in two subsection.

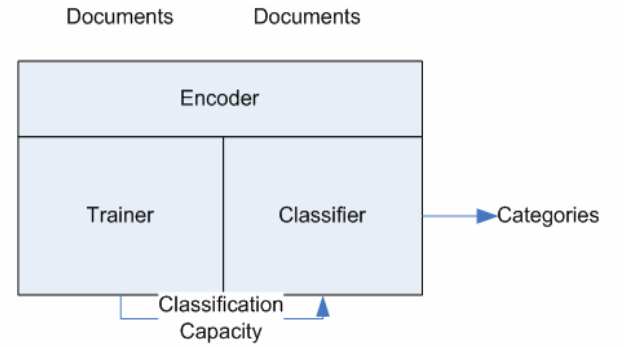

Fig. 4. Architecture of Text Categorization Systems

4.1. K-Means Algorithm

This subsection concerns a brief description of KNN in its traditional and modified versions. KNN is simple and practical as an approach to text categorization. It does not learn any training example until an unseen example is given. Therefore, KNN was called lazy based learning [13]. Since KNN depends strongly on labels of training examples for classify unseen examples, it is called examples based learning.

KNN is involved in the trainer and the classifier which are illustrated in figure 3. Sample encoded documents are given as input of the trainer. Before an unseen encoded document is given, sample encoded documents are only stored in the trainer. When an unseen document is given, the trainer is activated. The trainer receives the unseen document from the classifier, and ransfers its nearest training examples to the classifier.

The classifier implemented with KNN determines labels of unseen documents based on their nearest training examples. A number of nearest training examples is given as the parameter of KNN. The parameter is usually set as an odd number, such as one, three, five, and seven. When an unseen document is given, the classifier transfers it to the trainer. It receives its nearest training examples from the trainer and determines its label by voting target labels of the nearest training examples.

In this research, the two versions of KNN will be compared with each other. The first version is the traditional version where documents are encoded into numerical vectors which are used as input vectors. In the traditional version, two measures for computing a similarity between two numerical vectors are given. As expressed in equation (4), the first measure is the reverse of a distance between two numerical vectors.

$$x = [x_1, x_2, ..., x_d], y = [y_1, y_2, ..., y_d]$$
$$sim(x, y) = \frac{1}{\sqrt{\sum_{k=1}^{d}(x_k - y_k)^2}} \quad (4)$$

The second measure is cosine similarity as expressed in equation (5), and we will adopt this measure for implementing the traditional version of KNN.

$$x = [x_1, x_2, ..., x_d], y = [y_1, y_2, ..., y_d]$$
$$sim(x, y) = \frac{2x \cdot y}{\|x\| + \|y\|} = \frac{\sum_{k=1}^{d} x_k \times y_k}{\sqrt{\sum_{k=1}^{d} x_k^2} + \sqrt{\sum_{k=1}^{d} y_k^2}} \quad (5)$$

The second version of KNN is the proposed version where documents are encoded into string vectors as input data. The goal of the proposed version is to avoid the two main problems, huge dimensionality and sparse distribution, from the traditional version. A semantic similarity between two string vectors as expressed in equation (3) is used as the similarity measure between a training example and an unseen example in the proposed

version. A semantic similarity between two words given as elements is computed by fetching it from the similarity matrix. Note that it is necessary to build the similarity matrix before using this version of KNN for implementing the trainer and the classifier.

KNN is characterized by two properties. Its first property is that it does not learn any training example until any unseen example is given for its classification. Therefore, if too many training examples are given, it takes very much time for classifying unseen objects. Its second property is that it determines labels of unseen objects by referring target labels of involved training examples; it is called example based learning algorithm. Training examples relevant to a given unseen object becomes classification rules for it, in KNN.

5. Experiment and Results

This article concerns the experiments where two strategies of encoding documents for text categorization are compared with each other. We used the two test beds for these experiments: NewsPage.com and Reuter 21578. In order to compute an operation on string vectors, an inverted index of words are built from a corpus as the basis for doing that. We adopted KNN as the approaches to text categorization with their traditional and modified versions. The goal of these experiments is to observe whether modified version is comparable to their traditional versions, when we use the inverted index, instead of a restricted sized similarity matrix.

5.1. Experiment Data

This section concerns the two test beds used for these experiments. The first test bed is a small collection of news articles, called NewsPage.com. This test bed consists of five categories and totally 1,200 news articles. The second test bed is the standard collection of news articles, called Reuter21578. The test bed consists of more than one hundred categories and 21,578 news articles, and is popularly used for evaluating approaches to text categorization [16].

Table 1 illustrates the number of news articles in each category in the first test bed, NewsPage.com. There are totally 1,200 news articles which are exclusively labeled with one of five categories: ‘business’, ‘health’, ‘law’, ‘internet’, and ‘sports’. The source of this test bed is from the web site, www.newspage.com; the test bed is named after the URL address. We made the test bed as text files by copying and pasting full texts of news articles. In this test bed, each news article is given as an ASCII text file.

Table 2 illustrates the partition of the test bed into the training set and the test set. The task of text categorization on the test bed was decomposed into five binary classification tasks, according to the number of categories. In each binary classification task, there are two classes: positive class indicating that a document belongs to the corresponding category and negative class indicating that the document does not so. For each binary classification task corresponding to each category, we use F1-measure as the evaluation measure. As the general evaluation measure representing the five F1-meaures, we use micro-averaged and macro-averaged F1 measures.

Table 1. NewsPage.com

Category Name	#Document
Business	400
Health	200
Law	100
Internet	300
Sports	200
Total	1200

Table 2. Training Set and Test Set of Newspage.com

Categories	Training Set		Test Set	
	Positive Class	Negative Class	Positive Class	Negative Class
Business	280	280	120	120
Health	140	140	60	60
Law	70	70	30	30
Internet	210	210	90	90
Sports	140	140	60	60

The second test bed used for these experiments is Reuter21578. In this test bed, more than one hundred categories and 21578 news articles are given. We obtained the test bed by downloading it from the web site, http://www.daviddlewis.com/resources/testcollections/reuters21578/, and it has been popularly used as the standard test bed for evaluating approaches to text categorization [16]. In the previous test bed, news articles are exclusively labeled with only one category, while in this test bed, news articles are softly labeled with more than one category. News articles in this test bed are originally given as SGML files.

Table 3 illustrates the partition of this test bed into training set and test set for each category. Among more than one hundred categories, we select ten most frequent categories[6]. The text categorization task on this test bed is decomposed into ten binary classification tasks, according to the number of categories. In order to evaluate the approaches in the environment where a sparse number of training examples is given as an environment closer one to our reality, the number of training examples is restricted to 250, maximally. On this test bed, we use the identical measures for evaluating the approaches to those on the previous test bed.

[4] In the rest categories, each category has a very sparse number of news articles. Therefore, only ten most frequent categories are also selected as predefined ones in other literatures [Esbrooks et al 2004][Jo and Japkowicz 2004].

Table 3. Training Set and Test Set of Reuter21578

Categories	Training Set		Test Set	
	Positive Class	Negative Class	Positive Class	Negative Class
Acq	250	250	672	672
Corn	152	152	57	57
Crude	250	250	203	203
Earn	250	250	954	954
Grain	250	250	162	162
Interest	250	250	135	135
Money-Fx	250	250	246	246
Ship	176	176	87	87
Trade	250	250	160	160
Wheat	173	173	76	76

5.2. The Configurations of Involved Approaches

This section concerns the process of these experiments, together with the configurations of them. For the text categorization, documents are encoded into numerical vectors as large sized input data or string vectors as small sized ones. From training documents, an inverted index of words is built as the basis for the operation on string vectors in the modified version. KNN is adopted and used for these experiments as approaches to text categorization. This section specifies the configuration and process of these experiments.

Table 4 illustrates the configurations for these experiments, in the context of the two strategies of encoding documents. For using the traditional version, documents are encoded into large dimensional numerical vectors: 100, 250, or 500 dimensional numerical vectors. The inner product between two numerical vectors is used as an operation on them in the traditional versions. For using the modified version, documents are encoded into small dimensional string vectors: 10, 25, or 50 dimensional string vectors. The process of computing a semantic similarity between two string vectors is used as an operation on them for the modified version.

Table 4 illustrates the parameter settings for using the two supervised learning algorithms for text categorization. In KNN, the number of nearest neighbor given as its parameter is set to three.

Table 4. Configurations for these Experiments

K Nearest Neighbor	K = 3
Dimensions	Numerical Vectors: 100, 250, and 500 String Vectors: 10, 25, and 50

5.3. Experimental Results

This section concerns the results of comparing the two versions of KNN on the two test beds. Figure 5 and 6 illustrate the results as bar-graphs. In each figure, the y-axis indicates macro-averaged F1 in the left side and micro-averaged F1 in the right side. Within the x-axis, each group of bars indicates the traditional version or the modified version of either of the two supervised learning algorithms, and each individual bar within a group indicates a dimension of numerical vectors or string vectors into which documents are encoded. Among three-bars in each group, the white bar, the grey bar, and the black bar indicate a small dimension, a medium dimension, and a large dimension of numerical vectors or string vectors, respectively.

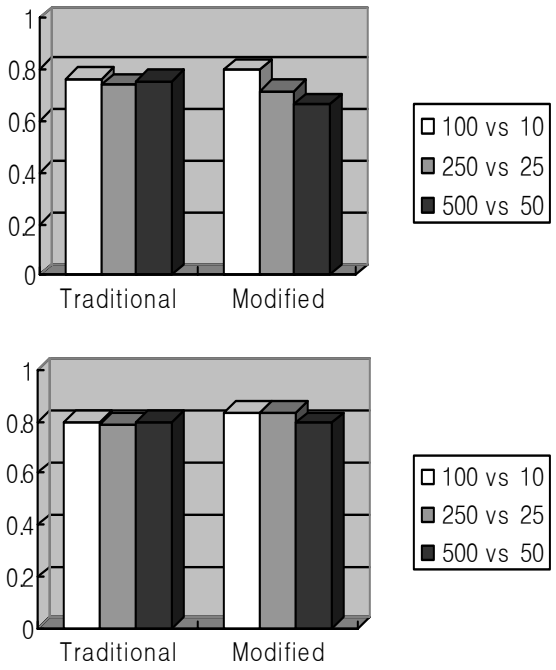

Fig. 5. The Results of Two Strategies of Encoding Documents in using KNN on NewsPage.com
Macroaveraged-F1 Measure (Top) and Microaveraged-F1 Measure (Bottom)

Figure 5 illustrates the results of comparing the two versions of KNN on the first bed called NewsPage.com. As illustrated in figure 5, in the traditional version, both macro and micro averaged F1 measures are close to 0.8, with regardless of dimensions of numerical vectors. In the modified version, macro-averaged F1 measures range between 0.6 and 0.8 as illustrated in the left side of figure 5, while micro-averaged F1 measures are around 0.8 as illustrated in the right side. With respect to macro-averaged F1 measures, the modified version is slightly better than the traditional version, only when documents are encoded into ten dimensional string vectors. With respect to micro-averaged F1 measures, the modified version is slightly better in all dimensions.

Figure 6 illustrates the results of comparing the two versions of KNN on the second bed called Reuter21578. As illustrated in figure 6, in the traditional version, its

macro-averaged F1 measures are around 0.75, while its micro-averaged F1 measures are around 0.6. In the modified version, both macro and micro averaged F1 measures are close to 0.8, with regardless of dimensions of string vectors. With respect to the macro-averaged F1 measures, both versions are close to each other, as illustrated in the left side of figure 6. With respect to the micro-averaged F1 measures, the modified version becomes clearly better than the traditional version as illustrated in the right side of figure 6.

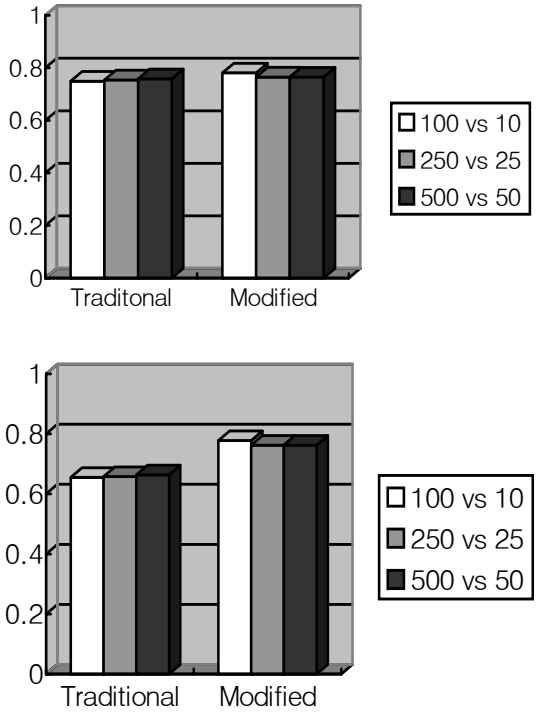

Fig. 6. The Results of Two Strategies of Encoding Documents in using KNN on Reuter21578
Macroaveraged-F1 Measure (Top) and Microaveraged F1 Measure (Bottom)

5.4. Discussion and Comments

Figure 7 visualizes the comparison of the traditional and modified versions of KNN spanning over the two test beds with respect to both kinds of F1 measures. In figure 7, the black part is the portion of the traditional version, while the white part is the portion of the proposed one. In the traditional version, the mean macro-averaged F1 measure which is averaged over dimensions and the two test beds is 0.7528, and the mean micro-averaged F1 measure is 0.7257. In the modified version, the mean macro-averaged F1 measure is 0.7475, and the mean micro-averaged F1 measure is 0.7937. With respect to the macro-averaged F1 measure, both versions of KNN are almost identical to each other. With respect to the micro-averaged F1 measure, the modified version is slightly better than the traditional version.

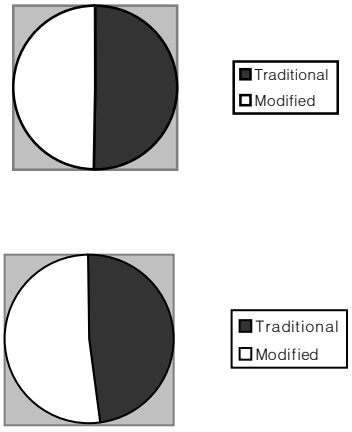

Fig. 7. The Comparison of Two Versions of KNN
Macroaveraged
F1 Measure (Top) and Microaveraged-F1 Measure (Bottom)

6. Conclusion

This research used a full inverted index as the basis for the operation on string vectors, instead of a restricted sized similarity matrix. It was cheaper to build an inverted index from a corpus than a similarity matrix, as mentioned in section 1. In the previous attempt, a restricted sized similarity matrix was used as the basis for the operation on string vectors. Therefore, information loss from the similarity matrix degraded the performance of modified version of KNN very much. This research addresses the information loss by using a full inverted index, instead of a restricted sized similarity matrix.

Note that there is trade-off between the two bases for the operation on string vectors. Although it is cheaper to build an inverted index from a corpus, note that it costs more time interactively for doing the operation expressed in equation (3). Let's the numbers of words, documents, and elements in each string vector be N, M, and d. In using the inverted index, the complexity for doing the operation is $O(M^2 d)$ in worst case, while in using the similarity matrix, the complexity is $O(d)$. When we try to compute semantic similarities of all possible pairs, the complexity is $O(N^2 M^2 d)$, whether we use a similarity matrix or an inverted index.

Other machine learning algorithms such as Naïve Bayes and back propagation are considered to be modified into their adaptable versions to string vectors. KNN 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.

References

[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] N. Cristianini and J. Shawe-Taylor, Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, 2000.

[3] H. Drucker, D. Wu, and V. N. Vapnik, "Support Vector Machines for Spam Categorization", IEEE Transaction on Neural Networks, Vol 10, No 5, pp1048-1054, 1999.

[4] A. Estabrooks, T. Jo, and N . Japkowicz, "A Multiple Resampling Method for Learning from Imbalanced Data Sets", Computational Intelligence, Vol 28, No 1, pp18-26, 2004.

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

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

[7] 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.

[8] T. Jo, and N. Japkowicz, "Class Imbalances versus Small Disjuncts", ACM SIGKDD Exploration Newsletters, Vol 6, No1, pp40-49, 2004.

[9] T. Jo and N. Japkowicz, "Text Clustering using NTSO", The Proceedings of IJCNN, pp558-563, 2005.

[10] R. R. Korfahage, Information Storage and Retrieval, Wiley Computer Publishing, 1997.

[11] 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.

[12] 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, pp59-65, 1992.

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

[14] 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.

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

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

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

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

Taeho Jo

He received BS, MS, and PhD from Korea University in 1994, from POSTECH in 1997, and from University of Ottawa in 2006, respectively. Currently, he is working for IT Convergence of KAIST as a senior research scientist. His research interests are Text Mining, Neural Networks, Machine Learning, and Information Retrieval. He has submitted and published almost 100 research papers since 1996. He has more than five years working experience in industrial organizations.