NTC (Neural Text Categorizer): Neural Network for Text Categorization

Taeho Jo
SITE, University of Ottawa
tjo018@site.uottawa.ca

Abstract

This research proposes a new neural network for text categorization which uses alternative representations of documents to numerical vectors. Since the proposed neural network is intended originally only for text categorization, it is called NTC (Neural Text Categorizer) in this research. Numerical vectors representing documents for tasks of text mining have inherently two main problems: huge dimensionality and sparse distribution. Although many various feature selection methods are developed to address the first problem, the reduced dimension remains still large. If the dimension is reduced excessively by a feature selection method, robustness of text categorization is degraded. Even if SVM (Support Vector Machine) is tolerable to huge dimensionality, it is not so to the second problem. The goal of this research is to address the two problems at same time by proposing a new representation of documents and a new neural network using the representation for its input vector.

1. Introduction

Text categorization refers to the process of assign a category or some categories among predefined ones to each document, automatically. Text categorization is a pattern classification task for text mining and necessary for efficient management of textual information systems. In the academic world, research on text categorization has been progressed very much, and we will survey it in next section. In the industrial world, text categorization systems were already developed as an independent system or a module for textual information systems [Jackson and Mouliner 2002]. Although research and development on text categorization have been progressed like this, we need further research on it to improve techniques and implementations of text categorization.

There are two types of approaches to text categorization: rule based and machine learning based approaches [Sebastiani 2002]. Rule based approaches mean ones where classification rules are defined manually in form of if-then-else, and documents are classified based on the rules. For example, classification rules are defined as, "business and company → company" meaning that if a document includes the two words 'business' and 'company', it is classified into the category, 'business' [Jackson and Mouliner 2002]. This class of approaches has high precision but poor recall, because of its poor flexibility. Machine learning based approaches mean ones where classification rules or equations are defined automatically using sample labeled documents. This class of approaches has a much higher recall but a slightly lower precision than rule based approaches. In addition to their poor flexibility, rule based approaches require time consuming manual jobs for building classification rules. Therefore, machine learning based approaches are replacing rule based ones for text categorization. This research focuses on machine learning based approaches to text categorization, discarding rule based ones.

Typical machine learning based approaches to text categorization are K Nearest Neighbor, Naïve Bayes, Support Vector Machine, and Back Propagation. They are used not only for text categorization, but also for any pattern classification problem, such as image classification, protein classification, and character recognition. Although there are other approaches than the five approaches, the four approaches are most typical and popular. In section 2, we will present previous cases of applying the four approaches to text categorization. In order to apply one of the four approaches to any pattern classification problem, raw data should be encoded into numerical.

Like any other pattern classification problem, in text categorization, it is true that documents given as raw data should be encoded into numerical vectors. The process will be described in detail in section 3. This strategy of encoding documents leads to two main problems: huge dimensionality and sparse distribution. In spite of using feature selection methods, a reduced dimension of numerical vectors representing documents still remains large. Excessive reduction of the dimension of numerical vectors using a feature selection method degrades the robustness of text categorization. The second problem, sparse distribution, leads to poor discrimination among numerical vectors for categorizing them. Although Support Vector Machine is very tolerant to huge dimensionality, it is not so to the second problem. Therefore, the goal of this research is to address the two problems at same time.

The idea of this research is to propose an alternative representation of documents to numerical vectors and a new supervised neural network as an approach to text categorization using the alternative representation in order to avoid the two problems. In this article, the alternative representation of documents is called string vector, and the proposed neural network is called NTC (Neural Text Categorizer). A sting vector is defined as a finite ordered set of words; it consists of words as its element, instead of numerical values. Since string vectors representing documents are classified robustly with their smaller dimension than numerical vectors in using the proposed neural network, string vectors are regarded as more compact representation of documents for text categorization. Additional advantage of string vectors is to provide more transparency in classification; it is possible to trace why documents are classified into such labels.

The architecture of NTC consists of three layers: input layer, learning layer, and output layer. Like Perceptron, the input layer is connected directly with the output layer, and the learning layer determines synaptic weights between the input layer and the output layer. The input layer corresponds to an input vector given as a string vector, and the learning layer and the output layer correspond to predefined categories. Each node in the learning layer has its own table consists of words and their weights indicating their membership of the corresponding category. Learning of NTC is the process of optimizing these weights in each table. NTC classifies unseen documents by computing output values by summing corresponding weights of string vectors.

The advantage of the proposed neural network is that NTC can classify documents with its sufficient robustness with its smaller input size and iterations of learning than traditional approaches using numerical vectors. Therefore, NTC solves the first problem, huge dimensionality, completely. Since sparse distribution can not exist in string vectors, the second problem is also addressed. Another advantage of NTC is that it provides transparency about its classification; it provides answer to why it classifies an unseen document into a particular category.

This article consists of six sections including this section. In section 2, we explore relevant previous research and consider its limitations in text categorization. In section 3, we describe in detail the process of encoding documents into numerical vectors and string vectors with the two subsections. In section 4, we describe the proposed neural network, called NTC, in detail, with respect to its architecture, learning process, and properties. In section 5, we compare the proposed neural network with other traditional approaches in text categorization, using three test beds. In section 6 as the conclusion, we will mention the significance of this work, and present directions of further research.

2. Previous Works

In this section, we will survey previous works relevant to this research, and point out their limitations. There exist other kinds of approaches to text categorization than machine learning based ones: heuristic and rule based approaches. Heuristic approaches were already applied to early commercial text categorization systems [Jackson and Mouliner 2000]. However, we count out the kind of approaches in our exploration, since they are rule of thumbs. Since rule based approaches have poor recall and require a time consuming job of building rules manually as mentioned in the previous section, they are not covered in this article, either. Therefore, this article counts only machine learning based approaches to text categorization considered as state of the art ones.

Typical machine learning algorithms applied traditionally to text categorization are KNN (K Nearest Neighbor), NB (Naïve Bayes), SVM (Support Vector Machine), and BP (Back Propagation). The four approaches to text categorization have been used more popularly in previous literatures on text categorization than any other traditional approaches. Among them, the simplest approach is KNN. KNN is a classification algorithm where objects are classified by voting several labeled training examples with their smallest distance from each object. KNN was initially applied to classification of news articles by Massand et al, in 1992 [Massand et al 1992]. Yang compared 12 approaches to text categorization with each other, and judged that KNN is one of recommendable approaches, in 1999 [Yang 1999]. KNN is evaluated as a simple and competitive algorithm with Support Vector Machine for implementing text categorization systems by Sebastiani in 2002 [Sebastiani 2002]. Its disadvantage is that KNN costs very much time for classifying objects, given a large number of training examples because it should select some of them by computing the distance of each test object with all of the training examples.

Another popular and traditional approach to text categorization is NB. Differently from KNN, it learns training examples in advance before given unseen examples. It classifies documents based on prior probabilities of categories and probabilities that attribute values belong to categories. The assumption that attributes are independent of each other underlies on this approach. Although this assumption violates the fact that attributes are dependent on each other, its performance is feasible in text categorization [Michell 1997]. Naïve Bayes is used popularly not only for text categorization, but also for any other classification problems, since its learning is fast and simple [Duda et al 2001].

In 1997, Mitchell presented a case of applying NB to text categorization in his textbook [Mitchell 1997]. He asserted that NB was a feasible approach to text categorization, although attributes of numerical vectors representing documents were dependent on each other; this fact contradicts with the assumption underlying in NB. In 1999, Mladenic and

Grobellink evaluated feature selection methods within the application of Naïve Bayes to text categorization [Mladenic and Grobelink 1999]. Their work implied that NB is one of standard and popular approaches to text categorization. Androutsopoulos et al adopted NB for implementing a spam mail filtering system as a real system based on text categorization in 2000 [Androutsopoulos, et al. 2000]. It requires encoding documents into numerical vectors for using NB to text categorization.

Another popular and traditional approach to text categorization is SVM. Recently, this machine learning algorithm becomes more popular than the two previous machine learning algorithms. Its idea is derived from a linear classifier, Perceptron, which is an early neural network. Since the neural network classifies objects by defining a hyper-plane as a boundary of classes, it is applicable to only linearly separable distribution of training examples. The idea of SVM is that if a distribution of training examples is not linearly separable, these examples are mapped into another space where their distribution is linearly separable, as illustrated in the left side of figure 1. SVM optimizes the weights of the inner products of training examples and its input vector, called Lagrange multipliers [Cristiani et al 2000], instead of those of its input vector, itself, as its learning process. It defines two hyper-planes as a boundary of two classes with a maximal margin, as illustrated in the left side of figure 1. Refer to [Hearest 1998] or [Cristiani et al 2000], for more detail description on SVM.

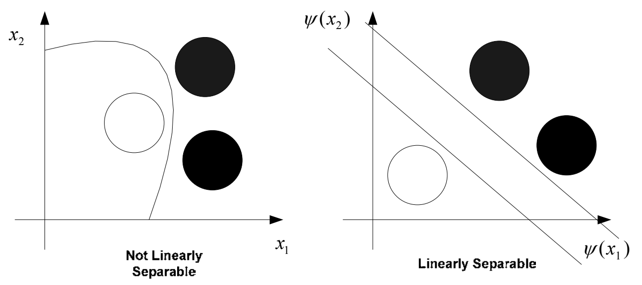

Figure 1. Mapping Vector Space in SVM

The advantage of SVM is that it is tolerant to huge dimensionality of numerical vectors; it addresses the first problem. Its advantage leads to make it very popular not only in text categorization, but also any other classification problems [Cristinani et al 2000]. In 1998, it was initially applied to text categorization by Joachims [Joachims 1998]. He validated the classification performance of SVM in text categorization by comparing it with KNN and NB. Drucker et al adopted SVM for implementing a spam mail filtering system and compared it with NB in implementing the system in 1999 [Drucker et al 1999]. They asserted empirically that SVM was the better approach to spam mail filtering than NB. In 2000, Cristianini and Shawe-Taylor presented a case of applying SVM to text categorization in their textbook [Cristianini and Shawe-Taylor 2000]. In 2002, Sebastiani asserted in his survey paper that SVM is most recommendable approach to text categorization by collecting experimental results on the comparison of SVM with other approaches from previous works [Sebastiani 2002]. In spite of the advantage of SVM, it has two demerits. One is that it is applicable to only binary classification; if a multiple classification problem is given, it should be decomposed into several binary classification problems for using SVM. The other is that it is fragile to the problem in representing documents into numerical vectors, sparse distribution, since the inner products of its input vector and training examples generates zero values very frequently.

The third popular and traditional approach to text categorization is BP. It is most popular supervised neural network and used for not only classification tasks but also nonlinear regression tasks [Haykin 1994][Hagan et al 1995]. It is also derived Perceptron, together with SVM. When a distribution of training examples is not linearly separable, in SVM, the given space is changed into another space where the distribution is linearly separable, whereas in back propagation, a quadratic boundary is defined by adding one more layer, called hidden layer [Haykin 1994][Hagan et al 1995], as illustrated in the right side of figure 1. More detail explanation about back propagation is included in [Haykin 1994] or [Hagan et al 1995].

In 1995, BP was initially applied to text categorization by Wiener in his master thesis [Wiener 1995]. He used Reuter 21578 as the test bed for evaluating the approach to text categorization and shown that back propagation is better than KNN in the context of classification performance. In 2002, Ruiz and Srinivasan applied continually back propagation to text categorization [Ruiz and Srinivasan 2002]. They used a hierarchical combination of BPs, called HME (Hierarchical Mixture of Experts), to text categorization, instead of a single BP. They compared HME of BPs with a flat combination of BPs, and observed that HME is the better combination of BPs. Since BP learns training examples very slowly, it is not practical, in spite of its broad applicability and high accuracy, for implementing a text categorization system where training time is critical.

Research on machine learning based approaches to text categorization has been progressed very much, and they have been surveyed and evaluated systematically. In 1999, Yang evaluated 12 approaches to text categorization including machine learning based approaches directly or indirectly in text categorization [Yang 1999][1]. She judged the three approaches, LLSF (Linear Least Square Fit), K Nearest Neighbor, and Perceptron, worked best for text categorization. In 2002, Sebastiani surveyed and evaluated more than ten machine learning based approaches to text categorization [Sebastiani 2002]. He asserted that Support Vector Machine is best approach to text categorization with respect to classification performance. All approaches which were surveyed and evaluated in these literatures require encoding documents into numerical vectors in spite of the two problems.

We explored and presented previous cases of applying one of the four traditional machine learning algorithms to text categorization. Although the traditional approaches are feasible to text categorization, they accompany with the two main problems from representing documents into numerical vectors. In the previous works, dimension of numerical vectors should reserve, at least, several hundreds for the robustness of text categorization systems. In order to mitigate the second problem, sparse distribution, a task of text categorization was decomposed into binary classification tasks in applying one of the traditional approaches. This requires classifiers as many as predefined categories, and each classifier judges whether an unseen document belongs to its corresponding category or not.

There is a previous trial to solve the two problems. In 2002, Lodhi et al proposed a string kernel for applying Support Vector Machine to text categorization [Lodhi et al 2002]. In

[1] In her study, direction evaluation means to evaluate approaches by performing experiments, while indirect evaluation means to evaluate them by collecting experimental results from other literatures.

their solution, documents as raw data are used directly for text categorization without representing them into numerical vectors. String kernel is a function computing an inner product between two documents given as two long strings. An additional advantage of the solution is to process documents independently of a natural language in which documents are written. However, their solution was not successful in that it took far more time for computing string kernel of two documents and the version of SVM using the string kernel was not better than the traditional version. As presented in section 5, this research will be a successful attempt to solve the two problems by proposing string vectors and a new neural network.

3. Document Representation

Since documents are unstructured data by themselves, they can not be processed directly by computers. They need to be encoded into structured data for processing them for text categorization. This section will describe the two strategies of encoding documents with the two subsections: the traditional strategy and the proposed strategy. The first subsection describes the former and points out its demerits, and the second subsection describes the latter and mentions its merits.

3.1. Numerical Vectors

A traditional strategy of encoding documents for tasks of text mining, such as text categorization is to represent them into numerical vectors. Since input vectors and weight vectors of traditional neural networks such as back propagation and RBF (Radial Basis Function) are given as numerical vectors, each document should be transformed into a numerical vector for using them for text categorization. Therefore, this subsection will describe the process of encoding documents into numerical vectors and what are their attributes and values.

Figure 2 illustrates the process of extracting feature candidates for numerical vectors from documents. If more than two documents are given as the input, all strings of documents are concatenated into a long string. The first step of this process is tokenization where the string is segmented into tokens by white space and punctuations. In the second step, each token is stemmed into its root form; for example, a verb in its past is transformed into its root form, and a noun in its plural form is transformed into its singular form. Words which function only grammatically with regardless of a content are called stop words [Frants et al 1997], and they correspond to articles, conjunctions, or pronouns. In the third step, stop words are removed for processing documents more efficiently and reliably for text categorization. Through the three steps illustrated in figure 2, a collection of words are generated as feature candidates.

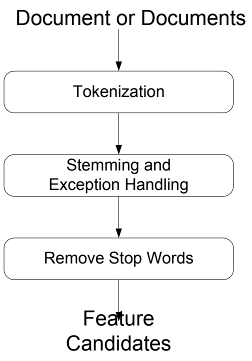

Figure 2. The process of encoding a document into a bag of words

Since the number of the generated feature candidates is usually too big, using all of them is not feasible as features of numerical vectors. Therefore, only some of them are used as features of numerical vectors for efficiency. A scheme of defining criteria for selecting some of them as features is called feature selection method [Mladenic and Grobelink 1999]. Generally, features are selected from the generated collection by their frequencies in the corpus. Therefore, candidates with highest frequencies are used as features of numerical vectors. The number of selected candidates as features becomes the dimension of numerical vectors. There are other feature selection methods than the frequency based one, and they are described in detail in [Mladenic and Grobelink 1999] and [Sebatiani 2002]. However, although only some of the candidates are used as features, the number of features is still large for robust text categorization[2].

The selected features are given as attributes of numerical vectors and numerical information about attributes become elements of numerical vectors. In this article, we mention the three ways of defining elements as the representative ones, although others may exist. The first way is to assign a binary value indicating absence or presence of the corresponding word in the given document; one indicates its presence and zero indicates its absence. The second way is to define elements as frequencies of corresponding words in the given document; the elements become integers which are greater than or equal to zero. The third way is to assign weights computed from equation (1) to elements of numerical vectors; elements are real values.

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

where $tf_i(w_k)$ is the frequency of the word, w_k, D is the total number of documents in the corpus, and $df(w_k)$ is the number of documents including the word, w_k in the given corpus. Note that the first and second way does not require the reference to a corpus, where as the third way requires the reference for computing elements of numerical vectors using equation (1).

Note that numerical vectors encoding documents have two main problems as mentioned in section 1. The first problem is that the dimension of numerical vectors is still large. This problems leads to high cost of time for processing each encoded document for

[2] Generally, several ten thousands feature candidates are generated from a particular corpus. Among them, several hundreds candidates are used as features. Therefore, the dimension of numerical vectors is several hundreds and is still high.

training a classifier and to requirement of a very large number of training examples proportionally to the dimension. The second problem is that each numerical vector includes zero values, dominantly. Since the discrimination among numerical vectors over categories is lost, categorization performance is degraded.

3.2. String Vectors

An alternative strategy of encoding documents for text categorization is to represent them into string vectors. In this subsection, we describe this strategy and its advantage in detail. However, this strategy is applicable to only NTC, while the previous one is applicable to any traditional machine learning algorithm.

A string vector is defined as a finite ordered set of words. In other words, a string vector is a vector whose elements are words, instead of numerical values. Note that a string vector is different from a bag of words, although both of them are similar as each other in their appearance. A bag of words is an infinite unordered set of words; the number of words is variable and they are independent of their positions. In string vectors, words are dependent on their positions as elements, since words correspond to their features. Features of string vectors will be described in detail in the next paragraph.

Features of string vectors are defined as properties of words to the given document. The features are classified into the three types: linguistic features, statistical features, and positional features. Linguistic features are features defined based on linguistic knowledge about words in the given document: the first or last noun, verb, and adjective, in a paragraph, title, or full text. Statistical features are features defined based statistical properties of words in the given documents; the highest frequent word and the highest weighted word using equation (1). Positional features are features defined based on positions of words in a paragraph or the full text: a random word in the first or last sentence or paragraph, or the full text. We can define features of string vectors by combining some of the three types, such as the first noun in the first sentence, the highest frequent noun in the first paragraph, and so on.

We can define features of string vectors in various ways as mentioned above, but in this work, features of string vectors are defined based on only frequencies of words for implementing easily and simply the module of encoding documents into string vectors. A d dimensional string vector consists of d words in the descending order of their frequencies in the given entire full text; the first element is the highest frequent word, the second element is the second highest frequent word, and the last element is the d the highest frequent word. Figure 3 illustrates the process of encoding a document into its string vector with the simple definition of features. In the first step of figure 3, a document is indexed into a list of words and their frequencies. Its detail process of the first step is illustrated in figure 2. If the dimension of string vectors is set to d, d highest frequent words are selected from the list, in the second step. In the third step, the selected words are sorted in the descending order of their frequencies. This ordered list of words becomes a string vector representing the document given as the input.

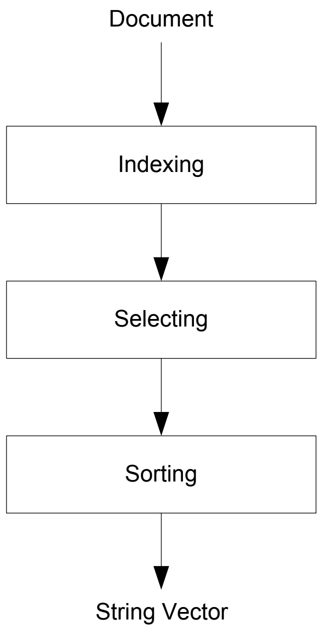

Figure 3. The process of mapping a bag of words into a string vector

This strategy of encoding documents for text categorization addresses the two main problems from the previous strategy. As presented in section 5, NTC using 50 dimensional string vectors is compared with other traditional approaches using 500 dimensional numerical vectors. The classification performance of NTC is comparable with the best traditional approach with much smaller input size and number of iterations. The experiments show that string vectors represent documents more compactly and efficiently than numerical vectors; the first problem is addressed. Since sparse distribution can not exist in string vectors, the second problem is also addressed.

Another advantage of string vectors is that string vectors represent documents more transparently than numerical vectors. Since each element of string vectors is symbolic data, it is possible to guess the content of the document by its surrogate; this is more user-friendly representation of documents than numerical vectors. Therefore, it is easier to trace why each unseen document is classified into a particular label in string vectors, than in numerical vectors.

4. NTC (Neural Text Categorizer)

This section describes the proposed neural network, NTC, in detail, with respect to its architecture, training, classification, and properties. The proposed neural network follows Perceptron in that synaptic weights are connected directly between the input layer and the output layer, and the weights are updated only when each training example is misclassified. However, note that NTC is different from Perceptron in context of its detail process of learning and classification, since it uses string vectors as its input vectors, instead of numerical vectors. The learning layer given as an additional layer to the input and the output layer is different from the hidden layer of back propagation with respect to its role. The learning layer determines synaptic weights between the input and the output layer by referring to the tables owned by learning nodes. The learning of NTC refers to the process of optimizing weights stored in the tables.

Figure 4 illustrates the architecture of the proposed neural network, NTC. It consists of the three layers: input layer, output layer, and learning layer. The input layer receives an input vector given as a string vector. The learning layer determines weights between the input and the output layer corresponding to words of the given input vector by looking up in the tables owned by learning nodes. The output layer generates the categorical scores indicating memberships of the string vector in categories as the output. The conditions of designing the proposed neural network, NTC, for text categorization are defined as follows.

- *The number of the input nodes should be identical to the dimension of string vectors representing documents.*

  This layer receives an input vector given as a string vector, so each node corresponds to each word in the string vector.

- *The number of the learning nodes should be identical to the number of predefined categories.*

  Nodes of this layer own tables corresponding to predefined categories, and determine weights between the input and output layer, to each word in the input vector.

- *The number of the output nodes should be identical to the number of predefined categories.*

  This layer generates categorical scores as the output, and they correspond to predefined categories.

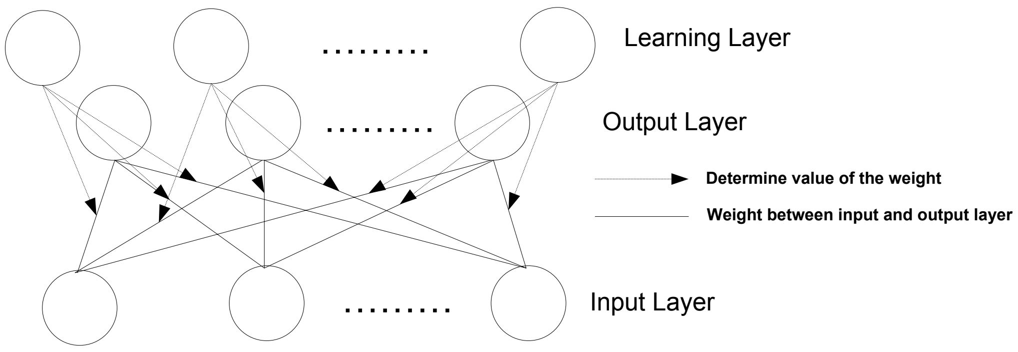
Learning Layer
Output Layer
Determine value of the weight
Weight between input and output layer
Input Layer

Figure 4. The Architecture of NTC

The first step of NTC is the initialization of weights which is the process of filling the tables which are empty initially. Each table corresponds to a predefined category, and it consists of entries. Each entry consists of a word and its weight. In this step, each weight is filled with the frequency of the corresponding word in the category corresponding to the table. Therefore, all tables owned by the learning nodes are constructed in this step.
The learning of NTC follows its initialization. An input vector given as a string vector is denoted by $\mathbf{x} = [t_1, t_2, ..., t_d]$, where t_i, $1 \le i \le d$, is a word given as an element of the string vector, $\mathbf{x}$, and d is the dimension of the string vector, $\mathbf{x}$. A set of the given predefined categories is denoted by $C = \lfloor c_1, c_2, ..., c_{|C|} \rfloor$. The weigh, w_{ji} denote the weight

connected between an input node, i, and an output node corresponding to the category, c_j $1 \le j \le |C|$. The value of the weight, w_{ji}, is defined, using equation (2),

$$w_{ji} = \begin{cases} table_j(t_i) & \text{if there is the word in the table} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $table_j$ denotes the table owned by the learning node corresponding to the category, c_j and $table_c(t_i)$ means the weights of the word, t_i, stored in the table, $table_j$. The weight, w_{ji}, means the membership of the word, t_i, in the category, c_j. Therefore, if there is the word, t_i, in the table, $table_j$, the weight, w_{ji}, is fetched from the table, $table_j$. Otherwise, the weight, w_{ji} becomes zero.

We compute the value of the output node, o_j, the output node corresponding to the category, c_j, using equation (3),

$$o_j = \sum_{i=1}^{d} w_{ji} \quad (3).$$

The value of o_j means the membership of the given input vector, $\mathbf{x}$ in the category, c_j. Since values of output nodes are combined by linear combination of weights illustrated in equation (3), the proposed neural network is similar as Perceptron. This is the first property shared with Perceptron.

As mentioned above, the learning of NTC is the process of optimizing weights between the input and output layer to minimize classification error in training examples. This learning is performed interactively to each training example. Each string vector in the training set has its own target label, c_j. If its classified category, c_k is identical to its target category, c, the weights does not change, as expressed in equation (4),

$$\text{if } c_j = c_k,\ \Delta w_{ki} = 0, \Delta w_{ji} = 0 \quad (4).$$

Otherwise, weights are adjusted to reinforce weights for its target category and to inhibit weights for its misclassified category, to minimize the classification error, as illustrated in equation (5),

$$\text{if } c_j \ne c_k,\ \Delta w_{ki} = -\eta w_{ki}, \Delta w_{ji} = \eta w_{ji} \quad (5)$$

where η is the learning rate given as a parameter, like any other neural networks, such as Perceptron, back propagation, and Kohonen Networks. This learning is repeated until the weights converge.

Figure 5 illustrates the process of learning sample documents and classifying unseen ones using NTC. A collection of sample labeled documents is given as the input, and the learning rate and the number of iterations are given as the parameters of NTC. In its first step, NTC initializes the weights stored in the tables owned by the learning nodes. For each sample labeled document, it is classified using equation (3) and the weights are updated using equation (5) whenever it is misclassified. This process is repeated with the fixed number, given as a parameter. After training NTC, unseen documents are classified by encoding them into string vectors, computing values of output nodes with the optimized weights using equation (3), and assigning the category corresponding to the output node with the highest value to each unseen document.

```
Classifier Training
Input: A Series of Sample Documents, Learning Rate, and Iteration Number
Step 1: Encode these sample documents into string vectors
Step 2: Design the architecture of NTC
Step 3: Initialize weights in each learning node with its document frequency within its corresponding category
Step 4: Repeat step 3-1 with the number of iteration
  Step 4-1: For each encoded sample document
    Step 4-1-1: Compute the values of output nodes of the encoded document with the current weight using the equation (3)
    Step 4-1-2: Classify each training string vector into the category corresponding to the output node with its highest value
    Step 4-1-3: If its classified category is different from its target category, update weights to every misclassification using the equation (5)
Output: Optimized weights in each learning node

Document Classification
Input: An unseen document and the optimized weights in each learning node
Step 1: Encode the unseen document into a string data
Step 2: Compute the values of output nodes of the encoded document with the current weight using the equation (5)
Step 3: Classify the unseen string vector into the category corresponding to the output node with its highest value
Output: its classified label
```

Figure 5. Process of training NTC and classifying unseen documents

Since NTC uses string vectors as its input vectors, the two main problems could be naturally avoided at same time. Each table owned by its corresponding learning node stores classification rules grained by training the NTC. These rules provide the basis of classifying documents more transparently than traditional machine learning algorithms using numerical vectors. Although string vectors used as input vectors in the proposed neural network address the two main problems, operations on string vectors are more restricted than those on numerical vectors. For example, we do not discover the method for finding a string vector representing a collection of string vectors, corresponding to a mean vector and a covariance matrix in numerical vectors. Therefore, NTC can not be trained in batch mode, because a mean vector can not be computed in string vectors.

5. Experimental Results

This section concerns experimental results of evaluating traditional and proposed approaches to text categorization on three test beds. In the experiments, five approaches, SVM, NB, KNN, Back Propagation, and NTC are evaluated as the approaches to text categorization, and three collections of news articles, Newspage.com, 20NewsGroups, and Reuter 21578, are used as the test beds of text categorization. In two of three test beds, the five approaches are evaluated both with decomposing text categorization into binary classification problems and without decomposing it.

In the experiments, documents are represented into string vectors for using NTC and numerical vectors for using the other methods. The dimensions of numerical vectors and string vectors representing documents are set as 500 and 50, respectively. In encoding documents into numerical vectors, most frequent 500 words from a given training set for each problem are selected as their features. The values of the features of numerical vectors are binary ones indicating the absence or presence of words in a given document;

this is for using Naïve Bayes. In encoding documents into string vectors, the most frequent 50 words are selected from a given document and sorted in the descending order of their frequencies as values of its corresponding string vector.

The parameters of the five approaches involved in this experiment are set by tuning them with a validation set, which is constructed by selecting 600 documents randomly from training documents, spanning the three test beds. Table 10 shows the definition of the parameters which is obtained through this tuning. With the parameters defined in table 10, the five approaches to text categorization will be applied to the three test beds.

Table 1. Parameters of the Five Approaches

Approaches to Text Categorization	Definition of Parameters
SVM	Capacity = 4.0
KNN	#nearest number = 3
NB	N/A
Back Propagation	Hidden Layer: 10 hidden nodes Learning rate: 0.3 #Iteration of Training: 1000
NTC	Learning rate: 0.3 #Iteration of Training: 100

5.1. NewsPage.com

The first set of this experiment pursues the evaluation of the five approaches on the test bed, Newspage.com, with and without the decomposition. This test bed consists of 1,200 news articles in the format of plain texts built by copying and pasting news articles manually and individually in the web site, www.newspage.com. Table 2 specifies 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 11, the ratio of training set to test set is set as 7:3. Here, this test bed is called Newspage.com, based on the web site, given as its source.

Table 2. 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

The task of text categorization on this test bed is decomposed into five binary classification problems, category by category. In each binary classification problem, a classifier answers whether an unseen document belongs to its corresponding category, or not. Table 3 shows the definition of training sets of the predefined categories. In table 3, ‘positive’ indicates that documents belong to the corresponding category and such documents will called positive documents, while 'negative' indicates that documents do not and such documents will be called negative documents. For each training set, all of documents not belonging to its corresponding category are allocated as negative documents. For each test set, negative documents are allocated as many as positive documents defined in the third column of table 2.

Table 3. The Allocation of Positive and Negative Class in Training Set of each Category

Category Name	Positive	Negative	Total
Business	280	560	840
Health	140	700	840
Law	70	770	840
Internet	210	630	840
Sports	140	700	840

Figure 6 presents the result of evaluating the five approaches on the test bed, Newspage.com, with a graph. On x-axis of the graph, the left group indicates the micro-averaged F1, the right group indicates the macro-averaged F1, and each bar within each group indicates one of the five approaches. The y-axis of the graph indicates the F1-measure which weight recall and precision, equally. The result of this evaluation shows that back propagation works best among the approaches with decomposition of the task of text categorization on this test bed into five binary classification problems. Although NTC is the second best approach to back propagation, it is comparable and competitive to back propagation, as shown in figure 6.

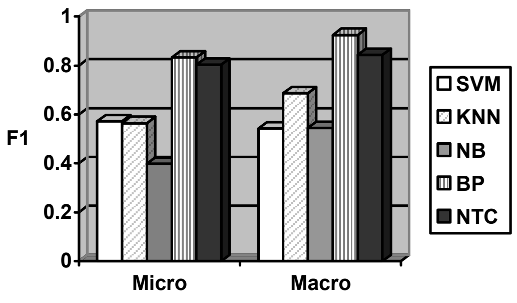

Figure 6. Result of evaluating five Text Classifiers in Newspage.com with decomposition

Figure 7 shows the result of evaluating the four classifiers except SVM without decomposition on this test bed. The reason of excluding SVM in this evaluation is that SVM is applicable only to a binary classification problem. Without decomposition, a classifier answers one of the five categories presented in table 11 and 12, instead of yes or no. Y-axis of figure 7 indicates accuracy which is the portion of correctly classified test documents to all of them, instead of F1-measure. This result shows that NTC is the best text classifier among the four approaches on this test bed without decomposition.

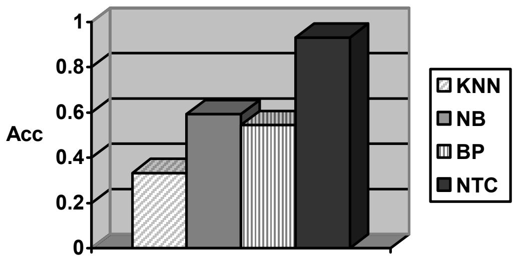

Figure 7. Result of evaluating four Text Classifiers in Newspage.com without decomposition

Although NTC is not better than back propagation in this test bed with respect to its performance, among the five approaches, NTC is preferable for implementing the module, 'classifier training' of DDO systems of the four-phase scenario, with two reasons. The first reason is that time taken for training a classifier is more critical than accuracy for the implementation. In the four-phase-scenario, training a classifier in the third phase leads to delay between creation mode and maintenance mode. During this period, an information system devotes itself to training a classifier. Although back propagation is a slightly better approach than NTC with respect to its performance, it takes time for training itself approximately fifty times that for training NTC. NTC is comparable and competitive with back propagation in spite of its tenth smaller dimension and iterations of training. The second reason is that NTC is more transparent than the others in classifying documents. For example, in back propagation, there is no way to find answer to the question, "why is an unseen document classified into a particular category?" Since NTC uses string vectors given as symbolic data as its input vector, it is possible to trace process of classifying unseen documents to answer the question. Whenever classifying an unseen document, we can show weights of elements given as words category by category to support why the document is classified into such a category.

5.2. 20NewsGroups

The second experiment is to evaluate the five approaches on another test bed, called '20NewsGroups'. This test bed is obtained by downloading it from the web site, http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html. This test bed consists of 20 categories and 20,000 documents; each category contains 1,000 documents. This test bed is partitioned into the training set and the test set with the ratio, 7:3; there are 700 training documents and 300 test documents per each category. Hence, 20,000 documents are partitioned into 14,000 training documents and 6000 test documents.

In this experiment, the task of text categorization on this test bed is decomposed into 20 binary classification problems, consistently with the number of predefined categories. A training set of each binary classification problem consists of 700 positive documents and 7000 negative documents. These negative documents are selected at random from 13,300 documents subtracted by 700 positive documents from 14,000 training documents. For a test set of each binary classification problem, 300 negative documents are allocated by selecting them randomly from 5,700 negative documents within the test set, in order to maintain the class balance in the test set.

Figure 8 shows the result of evaluating the five approaches on the test bed, 20NewsGroup. Since each category contain identical number of test documents, micro-averaged and macro-averaged F1 are same as each other. Therefore, their performances are presented in an integrated group, instead of two separated groups, in figure 8. This result shows that back propagation is also the best approach, while NB is the worst approach with the decomposition of the task on this test bed. Like the previous experiment set, NTC is comparable and competitive with back propagation.

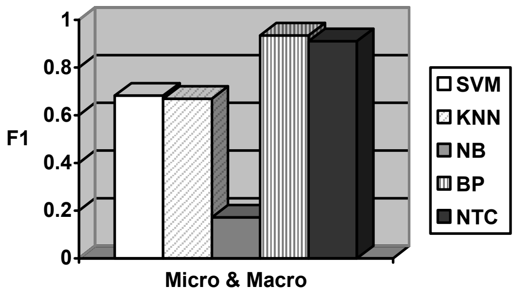

Figure 8. Result of evaluate the five text classifiers in 20Newsgroup with decomposition

Figure 9 shows the result of evaluating the four classifiers except the SVM without the decomposition on this test bed. In this case, a classifier answers to each test document by providing one of 20 categories. This result shows that there exits two groups: better group and worse group. The former contains back propagation and NTC, and the latter contains NB and KNN.

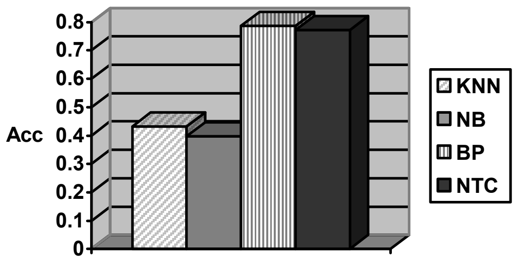

Figure 9. Result of evaluating four Text Classifiers in 20NewsGroups without decomposition

Like the previous set of this experiment, NTC is competitive with back propagation with smaller size of input data and lower number of training iterations. The result of this set is similar as that of the previous set, with respect to the trend.

5.3. Reuter 21578

The third experiment is to evaluate the five classifiers on the test bed, Reuter21578, which is a typical standard test bed in the field of text categorization. We selected most frequent ten categories instead of the entire categories. Table 4 shows labels of the ten selected categories and the number of training documents and test documents in each category. The partition of this test bed into training set and test set follows the version, ModApte, which is the standard partition of Reuter 21578 for evaluating text classifiers [Sebastiani 2002]. The number of documents in each category is very variable as shown in table 4. In this experiment, we can not evaluate these approaches without decomposition, since each document may have more than one category. Therefore, evaluation of these approaches without the decomposition was omitted in this set.

Table 4. Partition of Training Set and Test Set in 20NewsGroup

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

The task of text categorization on this test bed is decomposed into ten binary classification ones. For training set of each category, 2000 negative documents are allocated identically with regardless of the number of positive documents by selecting them at random from the remaining. For test set of each category, negative documents are allocated as many as positive documents, with its balance.

Figure 10 shows the result of evaluating the five approaches on this test bed with the decomposition. Unlike the two previous experiment sets, this result shows that NTC is the best approach among the five ones with respect to micro-averaged and macro-averaged F1. NTC has more difference from the others with respect to macro-averaged F1, as shown in the right side of figure 10.

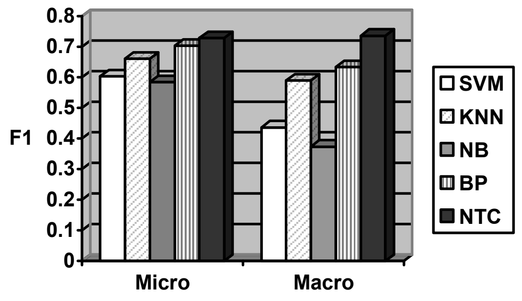

Figure 10. Result of evaluating five Text Classifiers in Reuter 21578 with decomposition

Figure 10 shows not only performance of the five approaches with respect to two evaluation measures, but also how much sensitive to sparse category containing a small number of positive training documents, these approaches are. More than half of ten categories correspond to sparse categories, as shown in table 4. Especially, SVM and NB show their large difference between their micro-averaged and macro-averaged F1, when the right side is compared with the left side of figure 10. This means that they are very sensitive to sparse categories. However, NTC shows little difference between two evaluation measures; this means that NTC is more tolerant to sparse categories.

This experiment implies that NTC is most practical among the five approaches. First, NTC has an acceptable performance in spite of its far smaller input size and learning iterations than those of the others. Although back propagation has the best classification performance in two of the three test beds, NTC is competitive and comparable to back propagation in its classification performance and is much faster than back propagation in its learning speed. Second, NTC is very tolerant to sparse categories, as mentioned above. In the practical world, such sparse categories may be given, very often. Based on this point, we may judge that NTC is more practical than the others in the real world.

6. Conclusion

This work addressed the two problems from representing documents into numerical vectors by proposing an alternative representation of documents and a new neural network using the representation. The proposed representation is called string vector, which is an ordered finite set of words. In other words, a string vector is a vector whose elements are words, instead of numerical values. The proposed neural network, called NTC, is based on Perceptron, with two points. The first point is that the input layer is connected directly with the output layer, and the value of each output node is computed by a linear combination of the weights. The second point is that the weights are updated only when a training example is misclassified. However, NTC is different from Perceptron, in that its input vectors are given as string vectors and its detail process of learning and classification is different from that of Perceptron, as illustrated in figure 4.

The experiments of the previous section presented that the proposed approach, NTC, is comparable with the best traditional approach, back propagation, with the smaller input size and number of iterations. This means that NTC is more practical than back propagation with respect to both the classification performance and the learning speed. Previous research has emphasized on only classification accuracy in competition of its own approach with other approaches. However, we must consider not only classification accuracy, but also other factors, such as learning speed and transparency for users, for evaluating practicality of approaches to text categorization. The significance of this research is that we proposed a practical approach, NTC, considering these factors at same time.

Although the current version of NTC was successful as an approach to text categorization, it treats string vectors as unordered sets of words. The process of initializing weights stored in the tables is independent of the order of string vectors. The three steps of NTC, initialization, learning, and classification, are independent of the order of string vectors. Since words in each string vector are sorted in the descending order of their frequencies, the first element is more important than the last element; elements should be discriminated for training NTC. The current version of NTC does not consider this situation. When features of string vectors are defined more sophisticatedly, the positions of elements of string vectors become more critical for text categorization. As a remaining task, we need to develop the next version of NTC which considers the order of elements of each string vector.

In this work, NTC was originally intended for only text categorization; it was applied to only text categorization in this work. Other traditional neural networks have been applied to not only text categorization, but also to other pattern classification problems. For example, back propagation is applicable to not only classification problems, but also regression problems. This fact is the strong point of back propagation. As a remaining task, we need to find other application areas where string vectors are better representation of raw data than numerical vectors and compare the proposed neural network with traditional ones in the tasks.

Literatures

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

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

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

Duda, R. O., Hart, P. E., and Stork, D. G., Pattern Classification, John Wiley & Sons, Inc, 2001.

Frants, V. I., Shapiro, J., and Voiskunskii, V. G., Automated Information Retrieval: Theory and Methods, Academic Press, 1997.

Hagan, M.T., Demuth, H.B., and Beale, M. Neural Network Design, PWS Publishing Company, 1995.

Haykin, S. Neural Networks: Comprehensive Foundation, Macmillan College Publishing Company, 1994.

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

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

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

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

Martin T. Hagan, Howard B. Demuth, and Mark Beale, Neural Network Design, PWS Publishing Company, 1995.

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

Mitchell, T. M., Machine Learning, McGraw-Hill, 1997.

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

Platt, J. C., “Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines”, Technical Report MSR-TR-98-14, 1998.

Rennie, J., “Improving multi-class text classification with support vector machine”, Master's thesis, Massachusetts Institute of Technology, 2001.

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

Sebastiani, F., “Machine Learning in Automated Text Categorization”, ACM Computing Survey, Vol 34, No 1, 2002, pp1-47.

Simon Haykin, Neural Networks: Comprehensive Foundation, Macmillan College Publishing Company, 1994.

Wiener, E. D., “A Neural Network Approach to Topic Spotting in Text”, The Thesis of Master of University of Colorado, 1995.

Yang, Y., “An evaluation of statistical approaches to text categorization”, Information Retrieval, Vol 1, No 1-2, 1999, pp67-88.