NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering

Taeho Jo
School of Information Technology and Engineering
University, Ottawa University
Ottawa, Canada

ABSTRACT: *This research proposes an alternative representation of documents and a new neural network using the proposed representation as its input and weight vectors, for more reliable text clustering. Almost traditional neural networks dictate representation of raw data into numerical vectors for their application to real tasks including text clustering. The traditional representation may lead to difficulties depend on the type of raw data. For example, in text clustering, encoding textual data given as raw data into numerical vectors leads to two main problems: huge dimensionality and sparse distribution. This research addresses the two problems at same time by proposing the alternative representation and a new neural network, called NTSO. The experiments in this research show that the proposed neural network is more practical than k-means algorithm, Kohonen Networks, and single pass algorithm, with respect to clustering performance and clustering speed.*

Keywords*:* Neural text self Organizer, String vector, Text clustering

Received: 11 June 2009, Revised 19 September, Accepted 8 October 2009

1. Introduction

Text clustering refers to a process of segmenting a collection of documents into sub-collections containing content based similar documents. Text clustering is an instance of pattern clustering, and unsupervised machine learning algorithms are traditionally used as approaches. Documents targeted for text clustering indicate electronic documents in a format of ASCII text, HTML, XML, or email, and have no their labels. In the practical world, initially, neither documents are labeled, nor are their categories predefined. Text categorization does not consider the reality; it is assumed that categories are clearly predefined, and sample labeled documents are given sufficiently in spite of this reality. To address this problem, text clustering and text categorization should be integrated with each other.

A typical and simple approach to text clustering is single pass algorithm. Single pass algorithm is suitable for implementing a real time text clustering system, since it clusters documents very fast; its complexity is almost linear to number of documents. The reason is that single pass algorithm does not optimize prototypes of clusters. In the initial step, the first cluster is created, and the first document is contained in the cluster. The document becomes the prototype of the first cluster. Note that similarity threshold is given as the parameter of this clustering algorithm, instead of the number of clusters. For each successive document, its similarity with the initial document[1] of each cluster is computed as one between the document and the cluster. We find the maximum similarity of the successive document and compare it with the similarity threshold. If the maximum similarity is greater than or equal to the similarity threshold, the document is included in the corresponding cluster[2]. Otherwise, one more cluster is created and the successive document is included in the cluster as its prototype. Since single pass algorithm does not optimize the prototypes of clusters, its cluster speed is high, but its cluster performance is not as high as Kohonen Networks and k-means algorithm.

[1]In another version, we can compute the similarity of a document with a cluster by averaging similarities with all contained documents [Vinokourov and Girolami 2000]. The reason of considering the only initial document as the prototype is to cluster document as fast as possible.
[2]The description of single pass algorithm is targeted for hard clustering. If it is soft clustering, the document is included in clusters whose similarity is greater than or equal to the similarity measure.

Another typical approach to text clustering is Kohonen Networks. In the previous text clustering system, called WEBSOM, the approach is adopted not only for text clustering, but also for word clustering [9]. In the next section, we will present previous cases of applying the approach to text clustering. Its architecture consists of two layers: competitive layer and input layer. Each node in the competitive layer corresponds to each cluster, and a winning node is selected through a competition of nodes. Synaptic weights between the two layers are connected and indicate prototype vectors of clusters. The learning of Kohonen Networks means the iterative optimization of the weights. Since this approach optimizes prototypes of clusters, its cluster performance is much better than single pass algorithm, but its clustering speed is not high.

K means algorithm is also a typical approach to not only text clustering but also any other pattern clustering. It is the simplest version of EM algorithm consisting of E-step and M-step [12]. Before explaining k means algorithm, we will describe briefly EM algorithm. In the EM algorithm, it is assumed that each cluster follows an identical distribution with its different parameters. Generally, the distribution is assumed to be a normal distribution characterized by mean vector and covariance matrix as its parameters. In the initial step of EM algorithm, parameters of clusters are initialized at random. Based on these parameters, probabilities that each object belongs to clusters are estimated in E-step. Based on these probabilities, the parameters are updated toward their maximum likelihood in M-step. EM algorithm clusters objects by repeating E-step and M-step alternatively until the parameters characterizing clusters converge.

In the k means algorithm, objects are selected as many as clusters at random. The selected objects become the initial prototypes of clusters. The random selection corresponds to the initial step of EM algorithm. The others are arranged into one of clusters based on their similarities with the prototypes. The arrangement corresponds to the E-step; probabilities that each object belongs to the clusters are given as a one and zeros. For each cluster, its mean vector is computed by averaging its contained vectors. In the k means algorithm, it is assumed that clusters follow normal distributions with different mean vectors and an identical covariance matrix. The identical covariance matrix is excluded in computing parameters, since it is independent of the probabilities. In the k means algorithm, the step of computing mean vectors of clusters corresponds to the M-step. Moving back to the E-step, the given objects are rearranged to clusters based on the mean vectors. There exist other clustering algorithms based on the EM algorithm than the k means algorithm. We will mention the previous research on the EM algorithm in the next section.

Although the three clustering algorithms are typical approaches to text clustering, note that documents should be represented into numerical vectors for using them as the approaches. The representation of documents for text clustering leads to the two main problems: huge dimensionality and sparse distribution. The former leads to high cost for processing each document for text clustering. The latter degrades clustering performance because discrimination among numerical vectors is lost. Therefore, we can not avoid the two problems if we use continually one of the three approaches for text clustering.

This paper proposes a new strategy of encoding documents for text clustering. This strategy is to represent documents into string vectors, instead of numerical vectors. A string vector is a finite ordered set of words; it is a feature vector whose elements are words, instead of numerical values. The proposed strategy is intended to address the two main problems in representing documents into numerical vectors.

This paper proposes also a new unsupervised neural network which uses string vectors as its input and weight vectors, as well as the strategy of encoding documents. The proposed neural network is called NTSO (Neural Text Self Organizer) in this article, and it follows Kohonen Networks in the context of the architecture and the learning rule. The architecture of NTSO consists of the competitive layer and the input layer, like Kohonen Networks. NTSO learns training examples by updating a weight vector corresponding to a winning node, iteratively. However, NTSO is different from Kohonen Networks with respect to detail computations involved in its learning, since the input vectors and the weights vectors are given as string vectors.

There are two traditional clustering algorithms with two extreme positions. In one position, there is Kohonen Networks which has good clustering performance but poor clustering speed. In the other position, there is single pass algorithm which is on contrary. NTSO is comparable with that of Kohonen Networks with respect to its clustering performance and scalability with its smaller input size and iteration number, as presented in section 6. NTSO is also comparable with single pass algorithm with respect to its clustering speed. Therefore, NTSO obtain advantages of the two clustering algorithms which locate in the two extreme positions.

This paper consists of seven sections including this section. Section 2 will present previous cases of applying single pass algorithm, Kohonen Networks, and EM algorithm to text clustering, to support the statement that the three algorithms are popular approaches to text clustering. Section 3 will describe the two strategies of encoding documents for text clustering. Section 4 will describe the proposed neural network, NTSO, in detail with respect to its architecture and learning process. Section 5 describes the proposed measure called clustering index, for evaluating text clustering systems. This measure will

be adopted for this evaluation in section 6, instead of F1 measure. Section 6 will present and discuss results of comparing the proposed neural network with the three traditional ones using two test beds. Section 7, as the conclusion, will mention the significance of this research, point out the demerit of the proposed neural network, and present solutions to the demerit.

2. Previous Works

This section explores previous cases of applying the traditional approaches described briefly in section 1. The traditional approaches will be compared with NTSO in text clustering, with respect to their clustering performance and speed in section 6. K means algorithm among EM algorithms is selected for the comparison with the proposed approach.

In 2000, Hatzivassiloglou et al defined two types of clustering algorithms: hierarchical and non-hierarchical clustering algorithms [8]. Hierarchical clustering algorithms are defined as algorithms whose parameter is similarity threshold and single pass algorithm corresponds to this class, whereas non-hierarchical clustering algorithms are defined as algorithms whose parameter is the number of clusters and Kohonen Networks, k-means algorithm, and NTSO correspond to this type [8]. They used additionally linguistic features for representing documents into numerical vectors for text clustering, and compared hierarchical clustering algorithms with each other. They shown empirically that single pass algorithm is the best approach with respect to its clustering speed and performance.

Kohonen Networks were created initially in 1982 by Kohonen as a self organizing algorithm of objects which are enabled to be represented into numerical vectors [10]. It was already described briefly in section 1. In 1998, Kaski et al implemented a text clustering system, called WEBSOM, where Kohonen Networks was adopted for clustering words and documents [9]. In the system, words and documents were clustered and a cluster of words was used as a label of a cluster of documents. In WEBSOM, each document was represented into a 315 dimensional numerical vector. Euclidean distance was used as the similarity measure between two numerical vectors in WEBSOM.

In 2000, Kohonen et al modified the WEBSOM which had been implemented in 1998 to improve its scalability; the modified version clusters a massive document collection with its much higher speed [11]. With respect to the process of clustering documents, both versions of WEBSOM are identical to each other. In the modified version, a hash table where identifiers of documents are given as keys is built, and it stores a winning node to each document. Instead of computing Euclidean distance between input vector and weight vector, the modified version retrieves the winning node from the hash table. Therefore, it clustered 6,840,567 patent abstracts with only 10% of time taken in the previous version, maintaining the clustering performance [11].

In 2002, Bote et al also adopted Kohonen Networks for implementing a text clustering system [3]. Note that their proposed text clustering system is different from WEBSOM, although Kohonen Networks was used for implementing both systems. In their proposed system each cluster was identified with a code consisting of numbers and uppercase alphabet letters, while in WEBSOM each cluster was identified with a group of words. The goal of the system is not to cluster unlabeled documents, but to map labeled documents topologically into a 20 by 20 matrix.

In 1977, Dempster et al proposed the EM algorithm, initially, as an iterative algorithm for estimating maximum likelihood of incomplete data [6]. Afterward, various versions of EM algorithm has been used as a clustering algorithm for generic objects [1] [4] and as an approach to text clustering. In 2000, Vinokourov and Girolami proposed five probabilistic models of hierarchical text clustering as specific versions of the EM algorithm [14]. EM algorithm was already described briefly in section 1. In 2003, Banerjee and his colleagues proposed two variants of the EM algorithm for soft clustering, where each object is allowed to belong to more than one cluster, and applied them to text clustering and gene expression clustering [2].

We explored the previous cases of applying the traditional approaches to text clustering. What is important in the previous research is that documents should be represented into numerical vectors for using one of the traditional approaches. We already mentioned the two main problems in the traditional strategy of encoding documents for text clustering. In spite of that, previous research has thought the two problems as unavoidable ones for clustering documents. Therefore, this research addresses the two problems by proposing an alternative strategy of encoding documents and a new unsupervised neural network coupled with the strategy.

3. Document representation

This section described two strategies of encoding documents for text clustering with two subsections. One is the traditional strategy, where documents are encoded into numerical vectors. The other is the proposed one, where documents are encoded into string vectors. In the first subsection, we describe in detail the traditional strategy, and mention its disadvantages. In the second subsection, we describe the proposed one, and mention its advantages.

3.1 Numerical Vectors

The traditional strategy of encoding documents for text clustering is to represent them into numerical vectors. Since unsupervised traditional neural networks, such as Kohonen Networks and ART (Adaptive Resonance Theory) receive numerical vectors as their input data, each document is required to be encoded into a numerical vector for using one of them for text clustering. This subsection describes the process of encoding documents into numerical vectors and attributes and their values of numerical vectors. In this subsection, we describe in detail the process of encoding documents into numerical vectors whose attributes are words.

For first, words are extracted as feature candidates from a particular corpus, and some of them are selected as attributes of numerical vectors. Figure 1 illustrates the process of extracting feature candidates from the corpus. All texts in the corpus are concatenated into a long string. In the first step, tokenization, the long string is segmented into tokens by a white space or a punctuation mark. In the second step, each token is stemmed into its root form; verbs in their past form are stemmed into their root form and nouns in their plural form are stemmed into their singular form. Here, words which perform only grammatical functions and are irrelevant to contents are called stop words, and conjunctions, articles, and prepositions correspond to stop words. In the third step, stop words are removed for processing documents for text clustering more efficiently. Through the three steps illustrated in figure 1, a list of words and their frequencies is generated as a group of feature candidates.

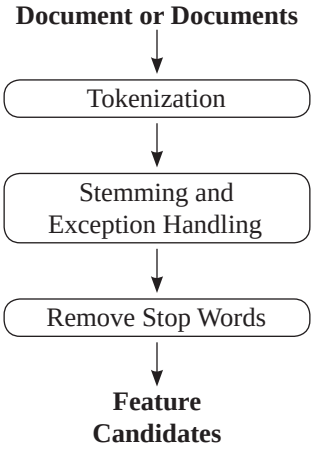

Figure 1. The process of encoding a document into a bag of words[3]

Since the number of feature candidates is usually huge, all of them are not feasible to use for features. Some of them are required to be selected as features of numerical vectors. For example, in the WEBSOM, more than 10,000 words are extracted as feature candidates [9]. The process of selecting some of words as features is called feature selection, and its scheme is called feature selection method. In this research, although there are many state of the art feature selection methods, frequency of feature candidates is used as criteria for selecting features, since it is simple and popular[4]. Although only some of feature candidates are used as features, many candidates should be selected for robust text clustering[5].

The selected words are given as attributes of numerical vectors representing documents. Elements of each numerical vector are numerical information about words given as features in the given document. The first way is to assign a binary value to each attribute of numerical vectors. The binary value indicates whether the corresponding word is present or absent in the document; one indicates its presence while zero indicates its absence. The second way is to define the frequency of the corresponding word in the given document as an element. In this way, elements become integers which are greater than or equal to zero. The third way is to define a weight of its corresponding word in the document as an element. The weight is computed using equation (1),

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

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. In the third way, elements of numerical vectors are given as

[3]Figure 1 was originally presented in [15].

[4]Stop words have their high frequency in general. But they were already excluded before generating feature candidates as illustrated in figure 1; stop words do not exist among feature candidates.

[5]Generally, several ten thousands feature candidates are generated from a corpus. We use only several hundreds candidates as features. However, the number of selected features is still large. For example, in the WEBSOM, 315 words are selected as features; each document is represented into 315 dimensional numerical vectors.

continuous real numbers. Elements defined in the first way and the second way are independent of the corpus, while elements defined in the third way are dependent on the corpus, since the variables dependent on the corpus, D and $df(w_k)$, are involved in computing elements.

Note that numerical vectors encoding documents have two main problems: huge dimensionality and sparse distribution, as mentioned in section 1. The influence of the problems on text clustering systems was already described in section 1. In the next subsection, we will describe an alternative strategy of encoding documents to solve the problems.

3.2 String Vectors

The alternative strategy of encoding documents for text clustering is to represent them into string vectors. In this subsection, we describe in detail the process of encoding documents into string vectors and advantages of the strategy. However, note that this strategy is applicable only to the proposed neural network, NTSO.

A string vector is defined as an ordered finite set of words. If we replace numerical values with words in numerical vectors, the vectors become string vectors. Note that a string vector is different from a bag of words, although both of them are similar as each other. A bag of words is an unordered infinite set of words; the number of words is variable and positions of words are meaningless. A position of each word in a string vector is significant since it corresponds to a feature.

Features of string vectors are defined as conditions of words in a given document, and their values are words which satisfy the corresponding properties. The conditions of words given as features of string vectors are classified into three types: linguistic features, statistical features, and positional features. Linguistic features indicate conditions based on grammatical functions of words. For example, the conditions such as first noun, first verb, and last noun in a particular sentence, paragraph, or an entire document, belong to linguistic features. Statistical features indicate conditions based on frequency, document frequency, and weight of words. The conditions such as the highest frequent word, the word with its highest weight, and the word with its lowest document frequency, belong to statistical features. Positional features indicate conditions based on positions of words in an entire document; the conditions, such as a random word in the first sentence and a random word in the last sentence, belong to positional features. We can define hybrid features of sting vectors by combining some of the three types. For example, the conditions, such as the noun in the first sentence, the highest frequent noun, and the highest frequent verb, belong to hybrid features.

Although features of string vectors could be defined variously as mentioned above, in this work, we define features of string vectors based on only frequency, in order to implement easily and simply a module of encoding documents into string vectors. A d dimensional string vector consists of d words sorted in the descending order of their frequencies; the first feature is 'the highest frequent word', the second feature is 'the second highest frequent word', and the last feature is 'the d th highest frequent word'. Figure 2 illustrates the process of converting each document into its corresponding string vector, using these features. In the first step, a document is indexed into a list of words and their frequency. The detail process of indexing a document or documents is illustrated in figure 1, and was already described in the previous subsection. In the second step, the d highest frequent words are selected from the words. In the last step, the selected words are sorted in the descending order of their frequency. Therefore, the process illustrated in figure 2 generates an ordered list of words, by their frequencies which become a string vector.

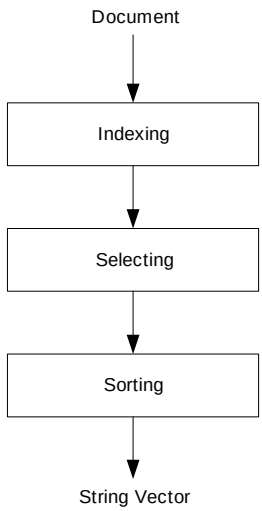

Figure 2. The process of mapping a bag of words into a string vector

In order to perform the operations on string vectors involved in training NTSO, we need to build a similarity matrix from a particular corpus. A similarity matrix is a word by word matrix and its elements indicate semantic similarities between words. Cristiani et al already proposed a word by word matrix indicating a similarity matrix by multiplying word-by-document matrix by a document-by-word matrix [5]. With two reasons, we propose another similarity matrix, instead of using Cristiani et al's one. First reason is that Cristiani et al's one defines semantic similarities as unlimited integers while the proposed one defines them as normalized continuous real values between zero and one. The second reason is that in Cristinani et al's similarity matrix, its diagonal elements are integers, while in the proposed similarity matrix, its diagonal elements are identically 1.0s.

A similarity matrix is denoted by

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

and its elements are computed by equation (2),

$$s_{ij} = sim(w_i, w_j) = \frac{\sum_{d_r \in D_i \cap D_j} (\phi_r(w_i) + \phi_r(w_j))}{\sum_{d_p \in D_i} \phi_p(w_i) + \sum_{d_q \in D_j} \phi_q(w_j)} \quad (2)$$

where its element S_{ij} indicates a semantic similarity between two words, w_i and w_j, and the function of a word, $\phi_p(w_i)$, indicates numerical information of the word, w_i in the document, d_p: a frequency, a weight computed by equation (1), or a binary value indicating its presence, and so on. Whatever numerical information we define as the function, $\phi_p(w_i)$, equation (2) means that the higher collocation of two words in their identical documents, the higher their semantic similarity.

The proposed similarity matrix has two inherent properties. The first property is that its diagonal elements are 1.0s. This means that it is assumed that two identical words have the highest semantic similarity, absolutely. The second property is that the similarity matrix is a symmetry matrix. This means that a semantic similarity between two words is identical with regardless of their order.

The two properties are expressed mathematically as follows, and proved based on equation (2).

Property 1. $S_{ij} = 1.0$ $(1 \le i \le N)$

Since two words at position, i are a word and itself, the column and the row are identical to each other and the similarity as computed with equation (32) is 1.0. Therefore the similarity matrix's its diagonal elements are 1.0.

<Proof>

$$s_{ii} = sim(s_i, s_i) = \frac{\sum_{d_r \in D_i \cap D_i} (\phi_r(w_i) + \phi_r(w_i))}{\sum_{d_p \in D_i} (\phi_p(w_i)) + \sum_{d_p \in D_i} (\phi_p(w_i))}$$

$$= \frac{\sum_{d_r \in D_i} (2\phi_r(w_i))}{2\sum_{d_p \in D_i} (\phi_p(w_i))} = \frac{2\sum_{d_r \in D_i} (\phi_r(w_i))}{2\sum_{d_r \in D_i} (\phi_r(w_i))} = 1.0$$

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

Since the operation of computing the similarity of two words, w_i and w_j using equation (32) is commutative, the similarity matrix is symmetric.

<***Proof***>

$$s_{ij} = sim(w_i, w_j) = \frac{\sum_{d_r \in D_i \cap D_j} (\phi_r(w_i) + \phi_r(w_j))}{\sum_{d_p \in D_i} \phi_p(w_i) + \sum_{d_q \in D_j} \phi_q(w_j)}$$

$$= \frac{\sum_{d_r \in D_i \cap D_j} (\phi_r(w_j) + \phi_r(w_i))}{\sum_{d_q \in D_j} \phi_q(w_j) + \sum_{d_p \in D_i} \phi_p(w_i)} = sim(w_j, w_i) = s_{ji}$$

Two documents denoted by two documents, d_i, and d_j, are encoded into two n dimensional string vectors [w_{i1}, w_{i2},…, w_{in}] and [w_{j1}, w_{j2},…, w_{jn}] , respectively by the process illustrated in figure 2. A semantic similarity between two string vectors is computed by equation (3),

$$sim(d_i, d_j) \approx sim(d_i^s, d_j^s) = \frac{1}{n} \sum_{k=1}^{n} sim(w_{ik}, w_{jk}) \tag{3}$$

From equation (3), we know that elements of the two string vectors correspond to each other 1: 1 link. The value of $sim(w_{ik}, w_{jk})$ is given as an element crossing the row corresponding to the word, w_{jk}, and the column corresponding to the word, w_{jk} in the similarity matrix; a similarity between two words is retrieved from the similarity matrix. The semantic similarity between the two string vectors is the average over similarities corresponding to 1:1 links of their elements. This operation corresponds to the inner product between two numerical vectors and is used for selecting a winning node of competitive layer of NTSO.

Another operation on string vectors for training NTSO is to define a set of inter-words between two input words. Inter-words are defined as words whose similarities with the two input words are greater than or equal to the similarity between the input words. For example, if the two input words are ‘computer’ and ‘hardware’, their inter-words become intuitively ‘CPU’, ‘mother-board’, ‘memory’, and so on. Equation (4) formally defines a set of inter-words of two input words.

$$\begin{aligned} I_{ij} = I(w_i, w_j) = \{ w_k \mid w_k \quad & s(w_i, w_k) \\ \geq s(w_i, w_j) \wedge s(w_j, w_k) & \geq s(w_j, w_i) \} \end{aligned} \tag{4}$$

Note that the set includes the input words, themselves. Since weight vectors of NTSO are given as string vectors along with input vectors, this operation is used for updating weight vectors for training NTSO.

Figure 3 illustrates the algorithm of building a set of inter-words of two input words. Two words, $word_i$ and $word_j$ are given as the input. The set of inter-words is initialized with the two input words, since the set includes the input words. If the two words are identical to each other, the algorithm returns the initialized set of inter-words. Otherwise, two rows of the given similarity matrix corresponding to the input words, $word_i$ and $word_j$, i th row and j th row are selected. Along these rows, i th row and j th row, if two elements are greater than or equal to the similarity between the two words, S_{ij}, the corresponding word is added to the set of inter-words. Finally, the algorithm illustrated in figure 3 returns a set of inter-words of the two input words, $word_i$ and $word_j$, as its output.

Input: two words ($word_i$ and $word_j$) and a similarity matrix
Process:
Inter Word Set ← {$word_i$,$word_j$}
If(word1 == word2), then return Inter Word Set
Similarity Threshold ← s_{ij} from the similarity matrix
For k = 1 to N
    If (s_{ik} >= s_{ij}) and (s_{jk} >= s_{ij}), then add $word_k$ to Inter Word Set
Return Inter Word Set
Output: Inter Word Set

Figure 3. Algorithm of building a set of inter words

String vectors have three advantages over numerical vectors for representing documents. The first advantage is that string vectors represent documents more compactly than numerical vectors. In other words, string vector represent documents with their smaller dimension enough for text clustering. In the experiments in section 6, NTSO using 50 dimensional string vectors is comparable to Kohonen Networks and k means algorithm using 500 dimensional numerical vectors, with respect to clustering performance. The first problem, huge dimensionality, is addressed by the proposed strategy. The second advantage is that string vectors don't have identical values dominantly. Numerical vectors representing documents tend to have zero values dominantly; this reduces their discrimination for text clustering. Therefore, the second problem, sparse distribution, is addressed in the proposed strategy. The third advantage is that string vectors are more transparent than numerical vectors. A string vector displays the content of its corresponding document for users. We can see the content of clusters by referring to their prototypes.

4. Neural Text Self Organizer

In this section, we describe in detail NTSO as the proposed approach to text clustering in the context of its architecture, learning process, and properties. Although the proposed neural network was originally intended only for text clustering, it may be useful actually for any clustering task where string vectors represent raw data more practically than numerical vectors. Although the proposed neural network follows Kohonen Networks with respect to its architecture and learning process, it is different essentially from the traditional neural network with respect to its detail computations, since weight vectors and input vectors are given as string vectors in the proposed neural network.

Figure 4 illustrates the architecture of the proposed neural network. The architecture consists of the competitive layer and the input layer; it is identical to the architecture of Kohonen Networks. The input layer receives an input vector given as a string vector; each node in the layer corresponds to a word given as an element, instead of a numerical value. The competitive layer selects a winning node through the competition of the contained nodes; each node corresponds to a cluster. The weight vectors between the two layers indicate prototypes of clusters, and are also given as string vectors, along with input vectors. Although NTSO has its identical architecture as illustrated in figure 4, it performs computations involved in learning differently from Kohonen Networks. Although the architecture illustrated in figure 4 is identical to that of Kohonen Networks, its detail process of learning training examples is different from that of Kohonen Networks.

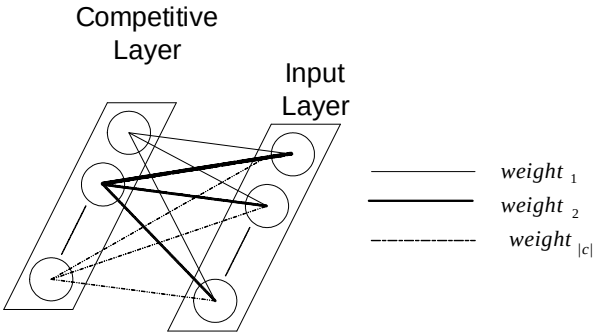

Figure 3. The architecture of NTSO

NTSO initializes its weight vectors differently from Kohonen Networks. Kohonen Networks initialize their weight vectors by randomizing numerical values, while the proposed neural network initializes its weight vectors using one of two strategies. The first strategy is to initialize weight vectors by selecting words from a particular corpus at random. The corpus for doing may be a particular one given separately or a collection of documents targeted for clustering. The second strategy is to assign some of string vectors to weight vectors at random. This strategy is similar as that of initializing prototypes in k means algorithm.

Like Kohonen Networks, NTSO selects a winning node in the competitive layer, and updates the weight vector originated from the node to the input nodes. However, note NTSO makes these behaviors differently from Kohonen Networks. A semantic similarity between an input vector and a weight vector for the competition for selecting a winning node in the competitive layer is computed using equation (3). Among the competitive nodes, the node from which the weight vector with the highest similarity as a given input vector is originated is selected as a winning node. Let's denote a *n*-dimensional input vector and the weight vector which is connected with the winning node by $input = [w_1, w_2, \cdots, w_n]$ and $weight_{\max} = [w_{\max 1}, w_{\max 2}, \cdots, w_{\max n}]$

, respectively. For each element of both vectors, we define a set of inter words, $I(w_{maxk}, w_k)$ $1 \le k \le n$ using the algorithm illustrated in figure 3. Here, we define a function of a particular set, rand(.), which generate a random element from the set. The weight vector is updated by replacing each of its elements by a random element of the corresponding set of inter-words, as expressed in equation (5).

$$\begin{aligned} &weight_{\max}(t-1) = [w_{\max 1}, w_{\max 2}, ..., w_{\max n}] \\ &input = [w_1, w_2, ..., w_n] \\ &weight_{\max}(t) \rightarrow [\text{rand}(I(w_{\max 1}, w_{j1})), \\ &\text{rand}(I(w_{\max 2}, w_{j2})), ..., \text{rand}(I(w_{\max n}, w_{jn}))] \end{aligned} \tag{5}$$

Figure 5 illustrates the process of clustering documents using the proposed neural network. In this process, a particular set of documents is given as the input and the number of clusters, dimension and the number of iterations are given as the parameters. Each document in the set is encoded into a string vector. The process of encoding it was already described in detail in section 3.2. The architecture of NTSO is set based on the number of clusters given as a parameter, and its weight vectors are initialized using one of the two ways. The architecture of NTSO is set based on the parameters, the number of clusters and dimension; the number of clusters determines the number of nodes in the competitive layer, and dimension determines that in the input layer. The weight vectors between the two layers are initialized with one of the two strategies, which are mentioned above. For each input vector, its similarities with weight vectors are computed, using equation (3). The weight vector with its highest similarity with the input vector is selected as the target for updating. The weight vector is updated using equation (5). To all documents, this process is repeated with the fixed number given as the parameter. For each input vector, its similarities with the optimized weight vectors are computed using equation (3). Therefore, it is arranged into the cluster corresponding to the weight vector with its highest similarity. Therefore, the process illustrated in figure 5 generates clusters of input vectors and their optimized prototypes as its output.

5. Experimental Results

This section concerns experiments for evaluating several approaches to text clustering. The four clustering algorithms, k means algorithm, single pass algorithm, Kohonen Networks, and NTSO, participate in this experiment. Two test beds where documents are exclusively labeled are used, in order to evaluate the four approaches easily.

The two different collections of news articles, 'NewsPage.com' and '20NewsGroups', are used in this preliminary experiment as test beds for evaluating the four approaches to text clustering. The collection, 'NewsPage.com', consists of 1200 news articles

Input and parameters: a list of documents, #clusters, dimension and #iteration
Step 1: Encode a list of documents into string vectors
Step 2: Set up the architecture of NTSO with the following conditions
    #input node = the dimension of each weight vector = the dimension of each numerical vector
    #competitive node = #weight vectors = #clusters
Step 3: initialize the given weight vectors by selecting words at random from the given corpus
Step 4: repeat step 4-1 #iteration times
Step 4-1: repeat the steps, 4-1-1 and 4-1-2 for each input vector
Step 4-1-1: compute similarities of weight vectors with the input vector as the values of the given competitive nodes using equation (3)
Step 4-1-2: update the weight vector with the highest similarity using equation (5)
Step 5: arrange each document into the cluster corresponding to the competitive node with its highest value
Output: a list of clusters including documents

Figure 5. The process of clustering documents using NTSO

labeled exclusively with one of five predefined categories. The collection, '20NewsGroups', consist of 20,000 news articles labeled exclusively with one of 20 categories. Among them, four categories are selected as representative ones for evaluating the four approaches to text clustering. For each test bed, ten clustering sets are build by selecting some of news articles at random. The numbers of documents are in the range between 100 and 1000 by incrementing 100 documents: 100, 200, 300, …., and 1,000.

The two different collections of news articles, 'NewsPage.com' and '20NewsGroups', are used in this preliminary experiment as test beds for evaluating the four approaches to text clustering. The collection, 'NewsPage.com', consists of 1200 news articles labeled exclusively with one of five predefined categories. The collection, '20NewsGroups', consist of 20,000 news articles labeled exclusively with one of 20 categories. Among them, four categories are selected as representative ones for evaluating the four approaches to text clustering. For each test bed, ten clustering sets are build by selecting some of news articles at random. The numbers of documents are in the range between 100 and 1000 by incrementing 100 documents: 100, 200, 300, …., and 1,000.

Table 1 illustrates the configurations of the parameters involved in the four approaches to the two test beds. The number of clusters, involved in k means algorithm, Kohonen Networks, and NTSO, is set consistently with the number of categories in each test bed: five for NewsPage.com and four for 20NewsGroups. Previous literatures proposing an approach to text clustering set the number of clusters given as its parameter, when evaluating its own approach and other approaches [8]. The similarity threshold given as a parameter to the single pass algorithm is set as 0.01 which results in the appropriate number of clusters. The single pass algorithm with this value of parameter, generates five clusters and four clusters in the first test bed and the second test bed, respectively. In this preliminary experiment, since the previous literature on text clustering and text categorization set input dimensions from 300 to 700 [9] [10], the dimension of numerical vectors is set as 500, by adopting a median among these input dimensions. Since the k-means algorithm and Kohonen Networks converged in around 500 iterations in the tuning phase in both test beds, the number of iterating updating prototype vectors is set as 500. Since NTSO started to converge in ten iterations and fifty iterations on NewsPage.com and 20NewsGroups respectively, the number of iterations for NTSO is set at ten and fifty for the first test bed and the second test bed respectively.

In this experiment, the clustering index, which is measure for evaluating clustering performance based on intra-cluster similarity and inter-cluster similarity, adopted as an evaluation measure for clustering algorithms. This measure was already described in the literature [16].

Clustering Algorithms	**Test Bed**	**Parameter Configurations**
K Means Algorithm	NewsPage.com	#clusters = 5 Input dimension = 500
	20NewsGroups	#clusters = 4 Input dimension = 500
Single Pass Algorithm	NewsPage.com	Similarity threshold = 0.01
	20NewsGroups	Input dimension = 500
Kohonen Networks	NewsPage.com	#clusters = 5 #iteration = 500 Input dimension = 500 Learning Rate = 0.2
	20NewsGroups	#clusters = 4 #iteration = 500 Input dimension = 500 Learning Rate = 0.2
NTSO	NewsPage.com	#clusters = 5 #iteration = 10 Input dimension = 50
	20NewsGroups	#clusters = 4 #iteration = 50 Input dimension = 50

Table 1. Definition of Parameters of the Four Clustering Algorithms

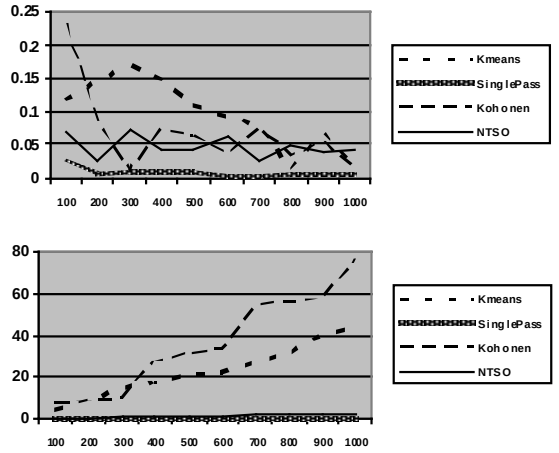

Figure 6. The Results of Evaluating Text Clustering Algorithms on NewsPage.com: Top (Clustering Index) and Bottom (Execution Time)

Figure 6 presents the results of evaluating the four approaches to text clustering with respect to clustering performance and speed on the first test bed, 'NewsPage.com'. The top line graph and the bottom line graph illustrate their clustering performance and clustering speed on the test bed, respectively. In the both line graphs, dotted line, dashed line, shaded line, and solid line indicate the trends of the k means algorithm, Kohonen Networks, single pass algorithm, and NTSO depending on the number of documents to cluster (size of the data set), respectively. In both line graphs, the x-axis indicates the number of documents to cluster. In the top line graph and the bottom line graph, the y-axis indicates the normalized value of the clustering index and the number of seconds taken for clustering the corresponding number of documents, respectively.

As illustrated in the top line graph of figure 6, it is shown that Kohonen Networks has the highest clustering performance entirely. The bottom line graph in figure 6 shows that the single pass algorithm and NTSO clusters document with highest speed, together, than the k means algorithm and Kohonen Networks. From the both line graphs, we can define two extreme positions. Kohonen Networks and k means algorithm, belong to a position, where clustering performance is good but clustering speed is poor. Single pass algorithm belongs to the opposite position. NTSO spans over the two positions; it has advantages of the two positions. With respect to scalability, NTSO is comparable with Kohonen Networks and k means algorithm in context of clustering performance, and is so with single pass algorithm in context of clustering speed, when we observe trends of the four approaches near 1000 documents in the both graphs.

Figure 7 illustrates the results of evaluating the four approaches with respect to the two factors on the second test bed, '20NewsGroups'. The top line graph presents that Kohonen Networks works best for text clustering with respect to clustering performance. It shows also that although NTSO is not as good as Kohonen Networks, it is competitive with k means

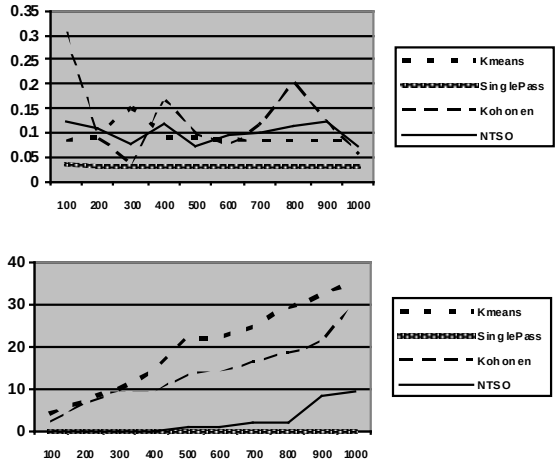

Figure 7. The Results of Evaluating Text Clustering Algorithms on 20 NewsGroups: Top (Clustering Index) and Bottom (Execution Time)

algorithm with respect to clustering index. According to the two line graphs contained in figure 7, we judge that NTSO spans over the two positions like the results illustrated in figure 6. When we observe the trends of the four approaches near 1000 document in the both graphs, NTSO is competitive with Kohonen Networks and k means algorithm with respect to clustering performance, and is so with single pass algorithm with respect to clustering speed.

6. Conclusion

This research proposed a new representation of documents, called string vector, and a new unsupervised neural network receiving the new representation as its input data, called NTSO, for text clustering. This research is intended for addressing two main problems in representing documents into numerical vectors for using traditional machine learning algorithms for text clustering: huge dimensionality and sparse distribution. In the previous section, the experiments show that string vectors represent documents more compactly than numerical vectors, maintaining the robust clustering performance; the first problem was addressed. Since string vectors have words as their elements, sparse distribution which may happen in numerical values including zero, never happen in string vectors; the second problem was also addressed. Although NTSO follows the conceptual fashion of Kohonen Networks in context of its architecture and learning rule, its computations involved in its learning are essentially different from those of Kohonen Networks.

Although the proposed neural network, NTSO, addresses the two problems completely, it has its own demerit which should be addressed in the next research. The demerit is that the proposed neural network depends on a similarity matrix; its computation on string vectors is impossible without defining a similarity matrix. Note that it costs very much time and system resource to build a similarity matrix from a corpus before applying the proposed neural network to text clustering. If we cluster again a collection of documents with the domain similar as that of previously clustered documents, we can reuse the previously built similarity matrix. Otherwise, we need to rebuild a similarity matrix from a new collection of documents. Therefore, the dependency of NTSO on a similarity matrix for the computations on string vectors should be addressed in a further research.

Although NTSO is intended initially for text clustering, it may be applied to any clustering problem where string vectors represent raw data more feasibly than numerical vectors. For example, image clustering using color information corresponds to the situation. In this problem, we can define a similarity matrix on color properties, manually easily, since there are only a small number of color properties. For example, we can define a high semantic similarity between 'light blue' and 'blue', and a low semantic similarity between 'yellow' and 'dark blue'. Selecting important pixels as features of string vectors representing images, we represent images into string vectors by filling the features with their corresponding colors, such as 'blue', 'yellow', and 'green'. Afterward we can use NTSO for image clustering.

References

[1] C. Ambroise and G. Govaert, Convergence of an EM-type algorithm for spatial clustering, Pattern Recognition Letters, v. 19, No 10, 1998, p. 919-927.

[2] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra, Generative model-based clustering of directional data, The Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, p19-28.

[3] G. Bote, P. Vincent, M. A. Felix, and V. H. Solana, Document Organization using Kohonen's Algorithm, Information Processing and Management, v. 38, No 1, 2002, p. 79-89.

[4] G. Celeux, and G. Govaert, A Classification EM algorithm for clustering and two stochastic versions, Computational Statistics & Data Analysis, v. 14, No 3, 1992, p. 315-332.

[5] N. Cristianini, J. Shawe-Taylor, and H. Lodhi, Latent Semantic Kernels, Journal of Intelligent Information Systems, v. 18, No 2-3, 2000, p. 127-152.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin, Maximum Likelihood from Incomplete Data via EM algorithm, Journal of the Royal Statistics Society, Series B, v. 39, No 1, 1977, p. 1-38.

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

[8] V. Hatzivassiloglou, L. Gravano, and A. Maganti, An Investigation of Linguistic Features and Clustering Algorithms for Topical Document Clustering, The Proceedings of 23rd SIGIR, 2000, p. 224-231.

[9] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen, WEBSOM-Self Organizing Maps of Document Collections, Neurocomputing, v. 21, 1998, p. 101-117.

[10] T. Kohonen, Self Organized Formation of Topologically Correct Feature Maps, Biological Cybernetics, v. 43, 1982, p. 59-69.

[11] T. Kohonen, S. Kaski, K. Lagus, J. Salojarvi, V. Paatero, and A. Saarela, Self Organization of a Massive Document Collection, IEEE Transaction on Neural Networks, v. 11, No 3, 2000, p. 574-585.
[12] T. M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
[13] P. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining*, Addison Wesely, 2006.
[14] A. Vinokourov, A. and M. Girolami, A Probabilistic Hierarchical Clustering Method for Organizing Collections of Text Documents, The Proceedings of 15th International Conference on Pattern Recognition, 2000, p. 182-185.
[15] Taeho Jo and Dongho Cho, Index Based Approach for Text Categorization, International Journal of Mathematics and Computers in Simulation, v. 2, No 1, 2007, p. 127-132.
[16] Taeho Jo and Malrey Lee, The Evaluation Measure of Text Clustering for the Variable Number of Clusters, Lecture Notes in Computer Science, v. 4492, 2007, p. 871-879.