DOI : 10.3745/JIPS.2008.4.2.067

Inverted Index based Modified Version of K-Means Algorithm for Text Clustering

Taeho Jo*

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

Keywords: *String Vector, K Means Algorithm, Text Clustering*

1. Introduction

Text clustering refers to the process of segmenting a group of piled documents into subgroups containing content based similar documents. In the task, a collection of piled documents is given as its input. The task generates a list of clusters containing similar documents in a flat or hierarchical form, as its output. Unsupervised learning algorithms, such as Kohonen Networks and EM algorithms, are used as state of the art approaches to text clustering. Text clustering is necessary for managing textual data for information systems automatically or semi-automatically.

It is more desirable to combine text clustering with text categorization, rather than to separate them from each other. Text clustering may be considered as a tool for automating preliminary tasks of text categorization. The first preliminary task for text categorization is to predefine a fixed number of categories. The second preliminary task is to allocate sample documents to each category. However, note that text clustering automates the preliminary tasks not by alone, but together with cluster identification [9].

There are various kinds of approaches to text clustering including heuristic ones [7]. In this research, the scope of approaches to text clustering is restricted to unsupervised learning algorithms. An unsupervised learning algorithm refers to a machine learning algorithm for clustering objects where unlabeled sample objects are given for training it. Typical unsupervised learning algorithms are Kohonen Networks, k means algorithm, and single pass algorithm. In this research, among them, we adopt k means algorithm and single pass algorithm as the targets for modifying into their adaptable versions to string vectors.

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

The second problem in representing documents into numerical vectors is sparse distribution. It refers to the phenomena where each numerical vector has dominantly zero values as its elements. This problem leads to poor discrimination among numerical vectors. The poor discrimination degrades both clustering and classification performance very much. In order to mitigate this problem, a given text categorization is decomposed to binary classification tasks as many as predefined categories in previous literatures [15][16].

The idea of this research is to propose an alternative strategy of encoding documents, in order to address the two problems. In the proposed strategy, documents are encoded into string vectors, and a string vector refers to a finite ordered set of words. In other words, a string vector has a finite number of words as its elements with their order, instead of numerical values. The goal of this

Manuscript received November 29, 2007; revised March 25, 2008; accepted May 6, 2008.
Corresponding Author: Taeho Jo

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

research is to address the two problems by representing documents into string vectors, instead of numerical vectors. An additional advantage of string vectors is that they are more transparent to users than numerical vectors; only a string vector enables users to guess the content of its corresponding document.

This research proposes the modified versions of k-means algorithm and single pass algorithm. In the modified versions, string vectors are used as their input data. In this research, we define an operation where a semantic similarity between two string vectors is computed. In the k-means algorithm, the operation is used for computing the similarity between a string vector and a representative one of a cluster. In the single pass algorithm, the operation is used for computing a similarity between a cluster and a document.

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

In this attempt, we will use an inverted index as the basis for the operation on string vectors involved in the modified version. An inverted index refers to a list of words each of which is linked to a list of documents including it. The advantage of an inverted index over a similarity matrix is that it is cheaper to build an inverted index from a corpus than a similarity matrix. The complexity of doing that reduces to $O(MN)$. Therefore, since it is possible to build a full inverted index from a corpus, in this research, it is expected to avoid the information loss from using a restricted sized similarity matrix.

This article consists of six sections including this section. In section 2, we will explore unsupervised learning algorithms which were previously applied to text clustering. The two strategies of encoding documents will be described in section 3, and the architecture of text clustering systems and two unsupervised learning algorithms will be described in section 4. In section 5, two versions of the k-means algorithm and the single pass algorithm will be compared with each other on two test beds. In section 6, as the conclusion of this article, the significance of this research and remaining tasks for future research will be mentioned.

2. Related Work

This article concerns the exploration of previous research on text clustering. As mentioned in section 1, there exist various kinds of approaches to text clustering. However, in exploring previous research, we restrict the scope of approaches only to unsupervised learning algorithms. Before exploring it, we will consider the process of computing a similarity between two documents, since the process is essentially necessary for text clustering. We explore previous cases of using Kohonen Networks, EM algorithm, and single pass algorithm for text clustering, as representative approaches, although there exist other machine learning based approaches.

What is most important for clustering objects including documents is to compute a similarity between two objects. In other words, clustering objects is strongly dependent on how to compute a similarity between two objects. Let's assume that objects are always represented into numerical vectors, $\mathbf{x} = [x_1 \quad x_2 \quad \dots \quad x_d]$ and $\mathbf{y} = [y_1 \quad y_2 \quad \dots \quad y_d]$. The first similarity measure is the reverse of a distance between two numerical vectors, as expressed in equation (1),

$$\begin{aligned} dist &= \|\mathbf{x} - \mathbf{y}\| = \sum_{k=1}^{d} \|x_k - y_k\| \\ similarity &= \frac{1}{dist} \end{aligned} \tag{1}$$

The second similarity measure is cosine similarity as expressed in equation (2),

$$\text{cosine similarity} = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} \tag{2}$$

A simple and popular clustering algorithm is single pass algorithm. When the number of clusters is far less than the number of objects, this algorithm runs in an almost linear complexity to the number of objects. This algorithm has been popularly used for clustering objects in industrial worlds, since it enables to implement a real time clustering system. However, note that quality of clustering objects in this algorithm is not as good as that in other clustering algorithms. In 2000, Hatzivassiloglou et al used this algorithm as an approach to text clustering where documents are encoded into numerical vectors together with linguistic features and compared it with complete pair-wise algorithm [6].

Kohonen Networks is an unsupervised neural network and was used as a popular approach to text clustering [3][14]. In this neural network, weight vectors are given as prototypes of clusters. Through its learning process, the weight vectors are optimized. Based on the similarity with the optimized weight vectors, objects are clustered. Because of the optimization of weight vectors, objects are clustered with higher quality in the clustering algorithm than in the single pass algorithm.

WEBSOM was a typical text clustering system where Kohonen Networks was adopted as the approach to text clustering [14]. In 1998, its initial version was developed by Kaski et al in 1998 [17]. Each cluster of documents is identified with a group of relevant words. In the system, not only documents, but also words are clustered using Kohonen Networks. Therefore, the system provides a visual organization of documents for browsing.

In 2000, Kohonen et al modified the WEBSOM which had been implemented in 1998 in order to improve its scalability; the modified version clusters a massive document collection with its much higher speed [14]. 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 winner to each input vector. Instead of computing Euclidean distance between input vector and weight vector, the modified version fetches the winner 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 [14].

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 [18]. EM algorithm is a paradigm of clustering objects where two main steps, E-step and M-step, are iterated for doing that. E-step refers to the process of estimating memberships of objects in each cluster, and M-step refers to that of estimating parameters representing distributions of clusters toward their maximum likelihoods. K-means algorithm will be described in detail in section 4 in the context of both its traditional version and its modified version.

Now, we explore previous research on the EM algorithm as a paradigm of clustering objects. In 1977, Dempster et al proposed the EM algorithm, initially, as an iterative algorithm for estimating maximum likelihood of incomplete data [5]. Afterward, various versions of EM algorithm has been used as clustering algorithms for generic objects [4] [1] and as approaches to text clustering. In 2000, Vinokourov and Girolami proposed five probabilistic models of hierarchical text clustering as specific versions of the EM algorithm [19]. In 2003, Banerjee et al 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].

In 2002, Lodhi et al attempted to solve the two main problems from representing documents into numerical vectors by proposing the string kernel for SVM [20]. The string kernel proposed by them is the operation on full texts where a syntactic similarity between two full texts is computed. Its additional advantage is that it is applicable independent of natural languages without considering their grammatical properties. Its disadvantage is that it takes too much time for performing the operation because of its very high complexity. Furthermore, their proposed version of SVM where the string kernel was used failed to be better than the traditional version of SVM.

In 2005, NTSO (Neural Text Self Organizer) was proposed as a solution to the two problems by Jo and Japkowicz [12]. NTSO is an unsupervised neural network which follows learning rule of Kohonen Networks and uses string vectors as its input vectors. Two operations are involved in training the neural network. The first operation is the process of computing a semantic similarity between two string vectors; it is also used in the proposed versions of SVM and KNN in this research. The second operation is the process of retrieving a set of inter-words between two words which are words with higher semantic similarities than that between the two words; the operation has very high complexity in NTSO.

3. Strategies of Encoding Documents

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.

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 [8]. 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. Although only some of feature candidates are used as features, many candidates should be selected for robust text clustering .

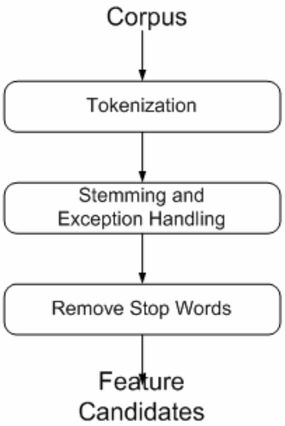

Fig. 1. The process of extracting feature candidates from a corpus

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),

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

where $weight_i(w_k)$ is the frequency of the word, w_k, $tf_i(w_k)$ 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 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.

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

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

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

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

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

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

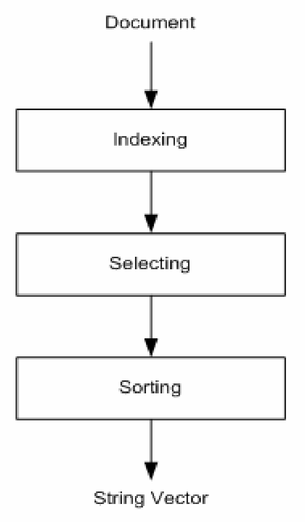

Fig. 2. The Process of Encoding Documents into String Vectors

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

A semantic similarity between two words is computed based on a number of documents where both words are collocated with each other. The more documents including both words, the higher semantic similarity between them is. From the inverted index, two lists of document identifiers corresponding to the two words are retrieved. The intersection is taken from the two lists of document identifiers as a list of documents including both words. Therefore, the semantic similarity is computed by equation (2),

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

where s_{ij} is a semantic similarity between the two words, w_i and w_j, $df(w_i)$ is a number of documents including the word in the corpus, w_i, and $df\,(w_i, w_j)$ is a number of documents including both words, w_i and w_j.

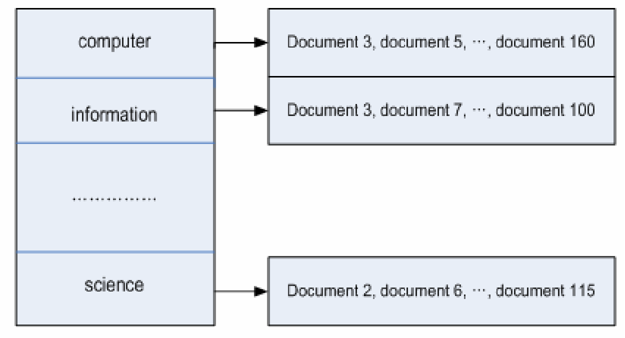
computer
Document 3, document 5, ···, document 160
information
Document 3, document 7, ···, document 100
……………
science
Document 2, document 6, ···, document 115

Fig. 3. Inverted Index

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

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

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

4. Text Clustering Systems

This section concerns architecture of text clustering systems and the modified version of k means algorithm. The architecture of text clustering systems is illustrated in figure 3. K-means algorithm is given as the engine of text clustering systems. In section 4.1, the traditional and proposed versions of k-means algorithm will be described briefly.

Figure 4 illustrates architecture of text clustering systems consisting of three modules: the encoder, the trainer, and the clusterer. The encoder encodes documents into numerical vectors or string vectors; the process was already described in section 3. The trainer optimizes prototypes of clusters by learning unlabeled encoded documents and transfers them to the clusterer, but it does not perform any function when the single pass algorithm is adopted as the approach to text clustering. The clusterer receives the optimized prototypes of clusters from the trainer, and arranges documents into the clusters based on the prototypes. The k-means algorithm is involved into the both modules, the trainer and the clusterer.

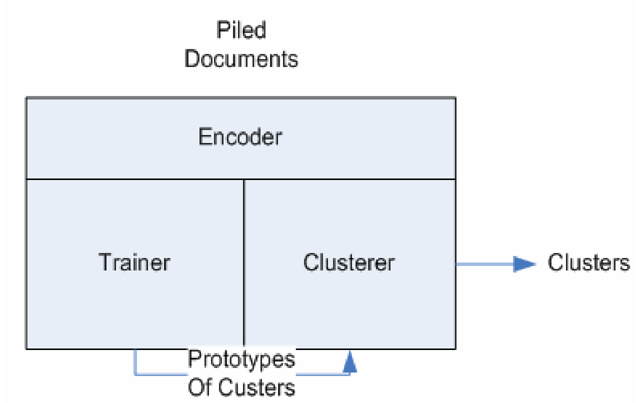

Fig. 4. Architecture of Text Clustering Systems

4.1 K-Means Algorithm

This section concerns the traditional and the proposed version of the k-means algorithm. The k-means algorithm has been most popularly used for clustering generic objects. The clustering algorithm was mentioned as the simplified version of EM algorithm [18]. Both versions of k-means algorithm learn unlabeled documents by iterating arranging documents and updating prototypes of clusters with a fixed number. As input vectors, the traditional version uses numerical vectors, while the modified on uses string vectors.

A number of clusters and a number of iterations are given as the parameters of the k-means algorithm. Since labeled documents are used for evaluating text clustering systems, the number of target labels becomes the number of clusters. However, the parameter is set arbitrary in real versions since the desirable number of clusters is not known in advance. When several versions of k-means algorithm are compared with each other, the second parameter may be set arbitrary. When the k-means algorithm is used for implementing a real text clustering system, the parameter may be replaced by convergence rate between previous prototypes and current ones.

Before applying the k-means algorithm to text clustering, their parameters should be determined. By selecting objects as many as clusters at random, the initial prototypes of clusters are determined. As the E-step (Estimation-Step), the others are arranged into clusters based on their similarities with the prototypes. As the M-step (Maximization-Step), by averaging objects in each cluster, prototypes are updated. Until the prototypes of clusters converge, the two steps are iterated.

For the modified version of this clustering algorithm, one more operation on string vectors is defined. The operation corresponds to the process of computing a mean vector which represents a group of numerical vectors. Figure 5 illustrates the operation on string vectors as the process of determining a string vector which represents a group of string vectors. If the number of d dimensional string vectors is m, the string vectors are represented into a m by d matrix of words as illustrated in figure 5. A representative string vector is built by selecting an element at random in each column of the matrix, as illustrated in figure 4.

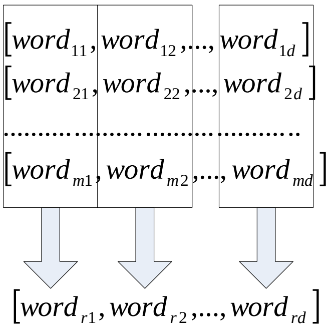
$[word_{11}, word_{12}, ..., word_{1d}]$
$[word_{21}, word_{22}, ..., word_{2d}]$
$[word_{m1}, word_{m2}, ..., word_{md}]$
$[word_{r1}, word_{r2}, ..., word_{rd}]$

Fig. 5. The Process of building Representative String Vector

In the modified version, the numbers of clusters and iterations are also determined in advance. Like the traditional version, by selecting string vectors randomly as many as clusters, initial prototypes of clusters are determined. As the E-step, the others are arranged into clusters which correspond to their most similar prototypes based on the criterion expressed in equation (5). As the M-step, prototypes of clusters are updated by running the process illustrated in figure 4. By iterating the E-step and the M-step with a fixed number, documents are also clustered in this version.

5. Experiment and Results

This article concerns the experiments where two strategies of encoding documents are compared with each other for text clustering. We used two test beds for these experiments: NewsPage.com and 20NewsGroups. In order to compute an operation on string vectors, inverted indices of words are built from a corpus as the basis for doing that. In these experiments, k means algorithm is adopted as the approaches to text clustering. The goal of these experiments is to observe whether modified version is comparable to their traditional versions, when we use the inverted indices as the basis for performing the operation on string vectors, instead of a restricted sized similarity matrix.

5.1 Experiment Data

This section concerns the two test beds used for these experiments. The first test bed is a small collection of news articles, called NewsPage.com. This test bed consists of five categories and totally 1,200 news articles. The second test bed is a large collection of news articles, called 20Newsgroups. The test bed consists of twenty categories and totally 20,000 news articles.

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

Table 1. NewsPage.com

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

Table 2 illustrates the five subgroups of news articles of this test bed for evaluating approaches to text clustering. Each subgroup consists of 500 news articles (100 news articles per category). News articles are given as ASCII text files named with sequential numbers in each category; each entry in table 2 consists of a number of documents identified from the start identifier to the end identifier in the given category. For example, a particular entry which is crossed by the column, ‘Subgroup 5’ and the row, ‘Heath’ consists of 100 and 101 ~ 200; the entry indicates that there are 100 news articles identified from 101 to 200 in the category, ‘health’ in the fifth subgroup. For each group, clustering index described in the first section is used as the evaluation measure, and mean clustering index is used as the overall evaluation measure.

Table 2. Five Sub-collections of NewsPage.com

Category Name	Subgroup 1	Subgroup 2	Subgroup 3	Subgroup 4	Subgroup 5
Business	100 1 ~ 100	100 51 ~ 150	100 101 ~ 200	100 151 ~ 250	100 201 ~ 300
Health	100 1 ~ 100	100 26 ~ 125	100 51 ~ 150	100 76 ~ 175	100 101 ~ 200
Law	100 1 ~ 100	100 1 ~ 100	100 1 ~ 100	100 1 ~ 100	100 1 ~ 100
Internet	100 1 ~ 100	100 51 ~ 150	100 101 ~ 200	100 151 ~ 250	100 201 ~ 300
Sports	100 1 ~ 100	100 26 ~ 125	100 51 ~ 150	100 76 ~ 175	100 101 ~ 200
Total	500	500	500	500	500

The second test bed is 20Newsgroup. The collection consists of 20 categories and 20,000 news articles. The test bed was obtained by downloading it from the web site, kdd.ics.uci.edu. Each news article is exclusively labeled with one of the twenty categories. This fact is the reason for adopting the collection as the test bed, instead of the most standard test bed, Reuter21578.

Ten subgroups are built from the test bed for evaluating the approaches to text clustering. The twenty predefined categories are grouped into two groups of ten categories. Each subgroup consists of ten categories and 500 news articles; each category in the subgroup contains 50 news articles. Among the ten subgroups of documents, the half spans over the first group of ten categories. The other half spans over the second group of the ten categories.

5.2 Experimental Process

This section concerns the process of the experiments and the configurations for them. For text clustering, documents are encoded into numerical vectors as large sized input data or string vectors as small sized ones. K means algorithm is adopted and used as the approaches to text clustering. From documents given as targets for clustering, inverted indices of words are built as the basis for performing an operation on string vectors in the modified versions of the two approaches.

Table 3 illustrates the configurations for these experiments, in the context of the two strategies of encoding documents. For using the traditional version, documents are encoded into large dimensional numerical vectors: 100, 250, or 500 dimensional numerical vectors. The inner product between two numerical vectors is used as an operation on them in the traditional versions. For using the modified version, documents are encoded into small dimensional string vectors: 10, 25, or 50 dimensional string vectors. The process of computing a semantic similarity between two string vectors is used as an operation on them for the modified version.

Table 3 also illustrates the configurations in context of the two approaches. In the configurations of k means algorithm, the number of clusters is set to five to the five subgroups of the first test bed and ten to the ten subgroups of the second test bed. In other words, the number of clusters is based on the numbers of categories of each subgroup in the both test beds.

Table 3. Configurations of this set of Experiments

		NewsPage.com	20NewsGroups
k-means algorithm	k	5	10
	#Iterations	50	
dimensions	Numerical vector	100, 250, and 500	
	String vector	10, 25, and 50	

5.3 Experimental Results

This section concerns the results of comparing the two versions of k means algorithm on the two test beds. Figure 6 and 7 illustrate the results of these experiments as bar-graphs. In each figure, y-axis indicates clustering index used as the measure for evaluating approaches to text clustering, and the clustering index was proposed by Jo and Lee in 2007 [11]. Within x-axis, each group of bars indicates the traditional version or the modified version of the two unsupervised learning algorithms, and each bar within a group indicates a dimension of numerical vectors or string vectors into which documents are encoded. Among three bars in each group, the white bar, the grey bar, and the black bar indicate a small dimension, a medium dimension, and a large dimension of numerical vectors or string vectors, respectively.

Figure 6 illustrates results of comparing the two versions of k means algorithm on the five subgroups illustrated in table 2. In the traditional version, when documents are encoded into 250 dimensional numerical vectors, the clustering performance is highest. In the modified version, when documents are encoded into ten dimensional string vectors, the clustering performance is highest. The maximum clustering indices of both versions are 0.0684 and 0.0586. The results indicate that the modified version catches up 85% with the traditional version with only tenth smaller input size.

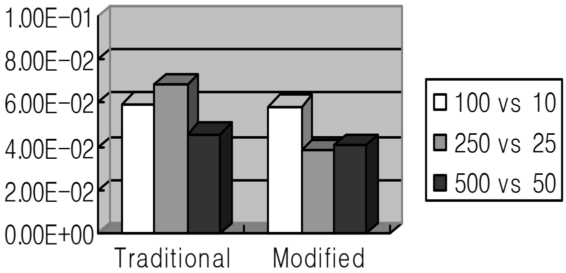

Fig. 6. The Results of two versions of k means algorithm on five subgroups in NewsPage.com

Figure 7 illustrates results of comparing the two versions of k means algorithm on the ten subgroups of the second test bed. Spanning over all dimensions, in the traditional version, the clustering index is around 0.02. In the modified version, the clustering index is also around 0.02. In the traditional version, clustering index is highest when documents are encoded into 500 dimensional numerical vectors, while in the modified version clustering index is similar in any dimension of string vectors. From the results illustrated in figure 7, it is judged that the modified version is comparable to the traditional version with the tenth smaller input size.

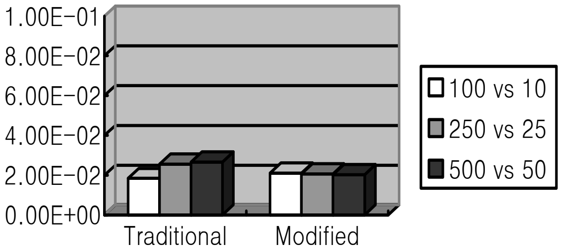

Fig. 7. The Results of two versions of k means algorithm on ten subgroups in Reuter21578

6. Conclusion

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

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

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

References

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

[2] 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, pp.19-28, 2003.

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

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

[5] 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, Vol.39, No.1, pp.1-38, 1977.

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

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

[8] T. Jo, "The Concepts of Text Mining", The Proceedings of ICACT 2000, pp.124-129, 2000.

[9] T. Jo, "Dynamic Document Organization using Text Categorization and Text Clustering, PhD Dissertation of University of Ottawa, 2006.

[10] T. Jo and M. Lee, "String Vectors in Unsupervised Learning for Text Clustering", Information Systems, submitted, 2007.

[11] T. Jo and M. Lee, "The Evaluation Measure of Text Clustering for the Variable Number of Clusters", Lecture Notes in Computer Science, Vol.4492 pp.871-879, 2007.

[12] T. Jo and N. Japkowicz, "Text Clustering using NTSO", The Proceedings of IJCNN, pp.558-563, 2005.

[13] T. Kohonen, "Self Organized Formation of Topologically Correct Feature Maps", Biological Cybernetics, Vol.43, pp.59-69, 1982.Kohonen, T., "Self Organized Formation of Topologically Correct Feature Maps, Biological Cybernetics, Vol.43, 1982, pp.59-69.

[14] 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, Vol.11, No.3, pp.574-585, 2002.

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

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

[17] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen, "WEBSOM-Self Organizing Maps of Document Collections", Neurocomputing, Vol.21, pp.101-117, 1998.

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

[19] A. Vinokourov, and M. Girolami, "A Probabilistic Hierarchical Clustering Method for Organizing

Collections of Text Documents", The Proceedings of 15th International Conference on Pattern Recognition, pp.182-185, 2000.

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

Taeho Jo

Taeho Jo received PhD degree from University of Ottawa in 2006, Master degree from POSTECH (Pohang Institute of Science and Technology) in 1997, and Bachelor from Korea University in 1994. Currently, he works for the School of Computer and Information Engineering at Inha University as a Professor. Previously, he had worked for Samsung SDS, ETRI (Electronic Telecommunication Research Insitute),. KISTI (Korea Insitute of Science and Technology Information), Nstein Technologies, BK21 Chonbuk National University, and IT Convergence Insitute for KAIST. He has published and submitted more than 100 research papers since 1996. His research interests are text mining, neural networks, machine learning, and information retrieval.