uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Taeho Jo

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

The Implementation of Dynamic Document Organization Using the Integration of Text Clustering and Text Categorization

TITRE DE LA THÈSE / TITLE OF THESIS

Nathalie Japowicz

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Nicholas Cercone (conference)

John Oommen

Diana Inkpen

Stan Matwin

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

The Implementation of Dynamic Document Organization using the Integration of Text Clustering and Text Categorization

Taeho Jo

Ottawa-Carleton Institute for Computer Science

School of Information Technology and Engineering (SITE)
University of Ottawa
Ottawa, Ontario, Canada

Your file Votre référence
ISBN: 978-0-494-18587-2
Our file Notre référence
ISBN: 978-0-494-18587-2

Table of Contents

List of Figures

List of Tables

Acknowledgement

I thank very much my academic supervisor, Professor Nathalie Japkowicz, Associate Professor, SITE (School of Information Technology and Engineering), University of Ottawa, for supervising and helping me for the research on this topic. She offered me the chance to research what I have pursued for my life. I appreciate her warm personality and good care during my PhD program. Really, I also thank her for helping my completion of PhD program, as soon as possible.

I thank my industrial supervisor, Mr. Arman Tajarobi, Director of Linguistic Engineering, Nstein Technology for supervising my working in the field for almost two years. Considering my residence in Ottawa and my progress in PhD program, he offered me flexible time for working for the company. Really, I appreciate his consideration and help for my convenience of my commute, working, and progress of my PhD program.

I thank the dissertation committee members, Professor Stan Matwin, Professor Diana Inkpen, and Professor John Oomen, for giving direction for improving PhD dissertation. I thank Professor Stan Matwin, Professor, SITE, University of Ottawa, for teaching data mining and machine learning necessary for the research of my PhD dissertation, and providing more previous literatures as background of this research. I thank Professor Diana Inken for having interest in my research and giving me direction of experiments. I thank Professor John Oomen for teaching pattern recognition necessary for this research and helping me, when I have insufficient oral communication skill.

I thank Professor Jeongyeon Shim, Assistant Professor, Kangnam University for giving me advice remotely on my life of graduate study as my senior, although she is in South Korea.

I thank my parents, my patents-in-law, and my brother-in-law for supporting me financially and morally from South Korea to continue my PhD study.

I thank my partner, Judy Kang, for managing my mind and helping me continually. She have picked me up for commuting between home and school, and provided meal for me. Although she is in South Korea, right now, I thank her for keeping touch with me.

Abstract

A document organization is a collection of documents composed of labeled clusters that contain similar documents. In any information system, a collection of documents always changes as time goes, since users access the collection to delete, add, and update documents. Dynamic Document Organization is a document organization that adapts automatically to such variable document collections. DDO poses two challenges, because of the decentralized mode of access. First, some clusters may have many documents, while others may have very few. Second, documents belonging to new topics may be added to the information system very often. Considering these two points, we need to reorganize the collection of documents, even if it was organized previously. Both text categorization and text clustering are limited when implementing DDO (Dynamic Document Organization) individually. Text categorization requires the manual preliminary tasks of the predefinition of a classification system and the preparation of sample labeled documents. Text clustering generates only unnamed clusters alone; each cluster should be labeled, manually by scanning contained documents. Therefore, this dissertation proposes approaches to the implementation of DDO that combined text clustering, cluster identification, and text categorization.

Chapter 1 Introduction

All information systems that organize and manage textual data, such as KMS (Knowledge Management System), IRS (Information Retrieval System), and DLS (Digital Library System), always have collections of documents that change continually as time goes. The reason is that users always delete, add, and update documents in these information systems. For example, if users of the systems add documents concerning a new issue continually, it is necessary to add a new category corresponding to such an issue. If a particular category grows to a big one from the continual addition of documents, it is necessary to split it into several categories. If a particular category reduces to a sparse one from the continual deletion of documents, it is necessary to merge it with its most similar category. The goal of this research is to develop techniques for managing variable collections of documents automatically and adaptively.

The first section of this chapter defines several key concepts related to the topic. Such concepts are *text categorization, text clustering, cluster identification, dynamic document organization,* and *resampling methods*. In the second section, we will explain the motivation for this research in order to explain why this topic was adopted. In the third section, we will describe the benefits from this research.

1.1 Basic Concepts

A *document organization* is a collection of documents composed of labeled clusters that contain similar documents. Note that a collection of non-clustered documents is not a document organization. If the document organization contains clusters with nested clusters, it is called a *hierarchical document organization.* If its clusters do not have any nested clusters, it is called a *flat document organization.* It is necessary to build a document organization, manually or automatically, for the efficient management of documents.

There are two types of document organizations, *static document organization* and *dynamic document organization.* If the clusters of the document organization are fixed permanently, it is called a *static document organization.* If it adapts by itself, to the current situation, we refer to the document organization as a *dynamic document organization.* Previous research on text categorization and text clustering has proposed state of the art approaches under the assumption of a static document organization. Since in the practical world the collection of documents always changes, such an assumption violates reality. Thus, it is necessary to assume a dynamic document organization in order to conduct more practical research about text clustering and text categorization.

Text categorization is the process of assigning one or several predefined and fixed categories to each text document as a special type of pattern classification. The techniques of text categorization are necessary for improving the quality of any information system dealing with textual data, although they cover only a fraction of document management techniques. There are two kinds of approaches for this task: rule based approaches and machine learning based ones [Sebastiani 2002]. In the former type of approaches, text documents are classified based on rules built manually, like expert systems, while in the latter type of approaches, they are classified based on classifiers trained with sample labeled documents. Recently machine learning based approaches have tended to replace rule based approaches, because machine learning based

approaches are more flexible. In spite of their advantage, machine learning approaches require a number of preliminary tasks for text categorization: predefinition of categories, manual labeling of sample documents, and the representation of documents into feature vectors. To use popular and traditional machine learning algorithms such as SVM (Support Vector Machine), Naïve Bayes, and Back Propagation, documents should be encoded into numerical vectors leading to two main problems: the large dimensionality of each feature vector and its sparse distribution.

Text clustering is the process of segmenting a group of documents into several clusters based on their similarity in content. Text clustering corresponds to unsupervised learning, while text categorization corresponds to supervised learning. The process of computing the similarity between two documents with respect to their contents is a very important task for text clustering; it influences the performance of text clustering. Text clustering generates a list of unlabeled clusters, as its output. Text clustering is too expensive a process for it to be performed frequently, although it does not require the kind of preliminary tasks required by text categorization.

Cluster identification is defined as the process of labeling the clusters generated from a text clustering. We stipulated that these names must satisfy three conditions, in order to automate the preliminary tasks for text categorization. The first condition is that two clusters are not allowed to have the same name, since there is no identical category in a classification system for text categorization. The second condition is that the name of each cluster must reflect the content of its component documents. The reason is that the category name is relevant to the contents of documents given as training examples for text categorization. The last condition is that the name of each category must not be too long. If so, it is difficult for users to read each category name at a time. These three conditions of labeling clusters should not be violated for browsing or generating sample documents for text categorization, as its preliminary tasks.

DDO (Dynamic Document Organization) is implemented using a circular integration of text clustering, cluster identification, and text categorization. As mentioned above, text clustering and cluster identification build a document organization automatically. The name of each cluster should satisfy the above three conditions. The classifiers are created using the documents included in the organization built using text clustering and cluster identification. Afterwards, these classifiers classify successive unseen documents and arrange them in the cluster corresponding to their classified category. If it is deemed that the organization after adding and deleting documents must be different from the current one, then the given documents are reorganized by transitioning from text categorization to text clustering. The rules for judging it are based on the number of transactions or the number of documents added and deleted in the system. DDO can then be seen as being implemented as a circular integration of text clustering, cluster identification, and text categorization.

Resampling is process of the manipulating the distribution of training examples of each class to maximize classification performance. The performance of classification depends not only on classification algorithms, but also on the distribution of training examples [Weiss 1998] [Weiss 2003]. Every machine learning algorithm is designed under the assumption that the distribution of training examples is completely balanced. In the practical world, training examples usually occur in a highly imbalanced distribution. An imbalanced distribution, however, degrades a classifier's performance toward the

minority classes, because of their strong bias on the majority class. The goal of resampling is to improve the performance of these classifiers by adjusting the imbalanced distribution of the training examples into a balanced one.

1.2 Motivation

The collection of documents in every information system always changes, since users keep on adding, deleting, and updating documents. Even though all the clusters may start with a balanced set of documents, as time goes, some clusters will have far more documents, while others may have far fewer. Overgrown categories need to be partitioned into several subcategories, while over-reduced categories need to be merged. Administrators of such information systems must redefine the structure of the categories and label the sample documents according to the updated categories, if the users' operations change the collection of documents. Such a task is very time consuming, since users must read a large number of electronic documents to label them individually.

For example, a collection of non clustered news articles is given on the left side of figure 1, and they are organized into three clusters labeled 'society', 'politics', and 'sports', as shown on the right side of figure 1. As illustrated on the left side of figure 2, news articles belonging to 'business', as a new topic, and 'sports', as an existing category, are added to the collection and news articles belonging to 'politics' are deleted from it. It is more desirable for news articles belonging to the new topic, 'business', to create a new cluster than to classify them into one of three existing clusters. In this example, the cluster, 'sports', corresponds to the overgrown cluster, and the cluster, 'politics', corresponds to the sparse one. It is more desirable to change the organization by splitting the overgrown cluster, 'sports', into two clusters, 'Olympics' and 'World Cup', and merging the sparse cluster, 'politics', with its most similar cluster, 'society', than to keep the existing organization. Through these actions on clusters, the new organization of news articles adaptable to the changed collection is shown on the right side of figure 2.

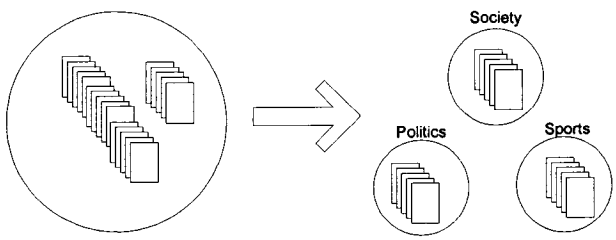

Figure 1. An Example of Document Organization

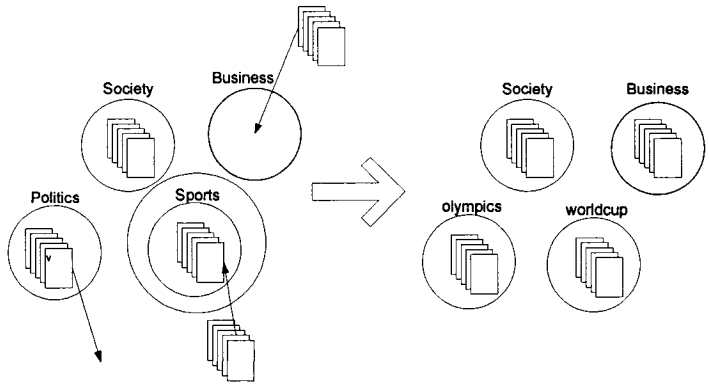

Figure 2. The Change of Collection from Addition and Deletion of Documents

Can text categorization be used to automate all tasks involved in document management in information systems? Although previous research proposed state of the art approaches to text categorization, such as SVM (*Support Vector Machine*), neural networks, k-nearest neighbor, Naïve Bayes, and decision trees, automatic text categorization covers only a small fraction of the tasks involved in document management. Once labeled sample documents are prepared and categories are predefined, an information system equipped with the module of text categorization classifies successive documents and arranges them into the location corresponding to the classified category automatically. With only text categorization, manual tasks, such as the predefinition of categories and the preparation of labeled sample documents, are required for information systems. Such tasks are very time consuming for a large collection of documents; labeled sample documents are very expensive to obtain in the practical world [Nigam, et al. 2000]. We can not guarantee that the predefinition of categories is fixed permanently; such manual tasks should be performed again whenever categories are changed. If such tasks take very much time, text categorization may become impractical.

Previous research proposed text clustering algorithms for automatic document organization [Mostafa and Lam 2000][Kohonen, et al. 2000][Hatzivassiloglou, et al. 2000][Kaski, et al. 2000][Wu, et al. 2001]. The results from text clustering only produce document organization with unnamed clusters. Furthermore, previous studies considered the task of text clustering only once under the assumption that a given document collection is permanent. It is not practical to cluster entire a document collection whenever a document is added or deleted. Furthermore, previous research on text clustering did not regard cluster identification as an important task involved in the preliminary task for text categorization. For example, M. Wu and his colleagues labeled clusters with only numbers [Wu et al. 2001]; this violates the second condition for access to the collection by browsing. The goal of this research is to make DDO more practical by integrating text clustering, cluster identification, and text categorization. Text clustering in DDO is performed only when it is necessary to reorganize documents because of the big difference between two organizations before and after adding and deleting documents.

Many studies have proposed machine learning based approaches including neural networks for text categorization and text clustering [Kohonen, et al. 2000][Kaski, et al. 2000][Wiener 1995][Ruiz and Srinivasan 2002]. When such approaches are applied to these two problems, each document should be represented into a numerical feature

vector. Such a representation causes two main problems; huge dimensionality of the feature vector and sparse distribution of each feature vector, which means that zero values are dominant within the vectors. The first problem lead to high cost for processing each document for text clustering or text categorization, and the second problem degrades the performance of the two tasks, because of the lack of discrimination among sparse numerical vectors. Although previous research proposed feature selection methods to reduce the dimension of feature vectors, there is a limit to the amount of reduction that can be achieved and furthermore they fail to solve the second problem, the sparse distribution of feature vectors; feature selection methods reduce the dimension of feature vectors from thousands to hundreds but such a reduction is not sufficient to be practical, since their dimensionality remains still high.

The distribution of training examples to classes is usually unbalanced in the practical world. Such a distribution is called *class imbalance* [Weiss 2003][Weiss 1998][Estabrooks, et al. 2004]. The class with fewer training examples is called the *minority class*, while the one with more examples is called the *majority class*. Class imbalance leads to a poor performance of classifiers on the minority classes because of their strong bias toward the majority classes. Resampling is one of the solutions that address this problem; it addresses the problem by adjusting the distribution of training examples toward near class balance. This research will apply resampling methods to the text categorization involved in implementing DDO.

1.3 Contributions of this Study

The first contribution of this research is to introduce the concept of managing documents completely automatically, and to study the paradigms of this new approach. Previous research has proposed techniques of text categorization or text clustering, intending originally to automate document management. However, its intention was changed to competitions of its own approaches with other approaches only in text categorization or text clustering, forgetting its original goal. This research pursues the goal of full automatic document management by integrating text clustering and text categorization with each other circularly, obtaining the original and practical goal.

The second contribution of this research is to propose an alternative representation of documents, called string vector, to numerical vectors. When we apply traditional machine learning algorithms to any problem of classification or clustering, raw data should be encoded into numerical vectors. For text categorization or text clustering, encoding documents given as raw data into numerical vectors leads to two main problems: huge dimensionality and sparse distribution. Therefore, this research addresses the two problems at same time by proposing the alternative representation of documents.

The third contribution of this research is to propose supervised and unsupervised neural networks using the alternative representation of documents as their input vectors. Once proposing the alternative representation of documents, we need machine learning algorithms using the alternative representation. This research proposes two machine learning algorithms; one is NTC (Neural Text Categorizer) as a supervised learning algorithm and the other is NTSO (Neural Text Self Organizer) as an unsupervised learning algorithm. This research will show that the proposed machine learning algorithms are more practical than traditional ones with respect to their performance and speed.

The fourth contribution of this research is to propose a new measure for evaluating text clustering systems and DDO systems. It is assumed that labeled documents are used as test bed for evaluating text clustering systems. The traditional measures, such as F1 measure, precision, and recall, are suitable not for text clustering, but for text categorization. The first reason is that when the single pass algorithm is used as an approach to clustering, the number of cluster is not predictable; the traditional measures require that the number of clusters should be fixed consistently with the number of predefined categories. Although the number of clusters is consistent with the number of predefined categories, results of clustering may be evaluated differently, depending on how to match clusters with predefined categories. Therefore, this research addresses this problem in evaluating text clustering systems, by proposing a new evaluation measure.
The last contribution of this research is to modify existing resampling methods to be adaptable to the proposed representation of documents. Almost all traditional resampling methods are targeted for the traditional representation of documents, numerical vectors. This research modifies these resampling methods for the alternative representation of documents and applies to training the proposed neural networks.

Chapter 2 Previous Research and Existing Approaches

In chapter 1, we discussed the basic concepts involved in this research, its motivation, and its contributions. This chapter surveys previous research on text clustering, cluster identification, text categorization, and resampling, since the first three topics are principally involved, and the last one is optionally involved in the implementation of DDO. Along with that, this chapter describes the existing strategies of document encoding, text clustering, and text categorization, and mentions the limits of the previous research and the expected advantages of this work.

This chapter consists of five sections. The first section describes the two existing strategies for encoding documents. The second section surveys previous research on text clustering, and describes the existing approaches: single pass algorithm, Kohonen Networks, and EM based algorithm. The third section surveys previous research on cluster identification, and discusses its difference between previous research and current research. The fourth section surveys previous research on text categorization, and describes the existing approaches: K Nearest Neighbor, Naïve Bayes, Support Vector Machine, and Back Propagation. The last section discusses class imbalance problem which occurs when classifiers are trained, and surveys previous research on resampling methods as solutions to the problem.

2.1 Existing Document Encoding

This section describes the two existing strategies for encoding documents and the process of computing a similarity between two encoded documents. *Document encoding* is defined as the process of representing documents into structured data. The reason for document encoding is that documents are themselves, unstructured data on which text categorization can not be applied directly. Therefore, the two subsections discuss the two document encoding techniques currently in use – bags of words and numerical vectors— along with their associated similarity measures.

There are three types of features used in bags of words or numerical vector representations: uni-gram, bi-gram, and n-gram. A uni-gram refers to a term consisting of a single token. 'Company', 'customer', 'service', and 'business' are all examples of uni-grams. A bi-gram refers to a term consisting of two tokens such as 'economy status', 'artificial intelligence', 'computer architecture', and 'computer system'. A n-gram refers to a term consisting of three tokens or more, such as 'text categorization system', 'text clustering software', 'English as a second language course', and 'neural processing information system'. Since not only uni-grams but also bi-grams and n-grams can be given as elements of bags of words or attributes of numerical vectors, an n-gram, itself, should not be confused for the representation of an entire document; it is not an encoded document like a bag of words, or a numerical vector.

2.1.1 A Bag of Words

The simplest way of encoding documents is to represent them into bags of words by indexing them. A bag of words is a set of words which is unordered and variable in its size depending on the length of the given document, and denoted by $d_i^b = \{w_{i1}, w_{i2}, ..., w_{il}\}$, where i is an index of a particular document and l is the size of

the set, d_i^b. This section describes the process of encoding a document into a bag of words and computing a similarity between two documents represented into two bags of words.

Figure 3 presents the process of encoding a document into a bag of words. Through the first step, tokenization, a text is segmented into tokens by white space. Each token is then converted into its root form through the second step, stemming and exception handling. For example, a plural noun is converted into a singular form and a verb in past form is changed into its root form. The last step removes stop words, which perform just a grammatical function regardless of the content of the text, and stop words include preposition, conjunction, particle, auxiliary verbs, and so on. The final output of this process is an unordered list of words representing a document, as illustrated in figure 3.

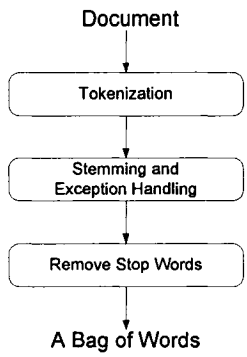

Figure 3. The process of encoding a document into a bag of words

Two documents, d_i and d_j, are encoded into two bags of words denoted by $d_i^b = \{w_{i1}, w_{i2}, ..., w_{il}\}$ and $d_j^b = \{w_{j1}, w_{j2}, ..., w_{jm}\}$. The numbers of words in the two sets, l and m, are different from each other in general. The similarity between these two documents, d_i and d_j is computed by one of the three equations, equation (1), (2), and (3). The equations are derived from equation (5), (6), and (7), respectively, in order to compute a similarity between two bags of words, instead of numerical vectors. In equation (1), the similarity is the ratio of twice the number of shared words, which is the size of the intersection of two sets, to the addition of the size of each set. In equation (2), it is the ratio of the size of the intersection of two sets to that of the union of them. In equation (3), it is the average of the ratios of the size of the intersection of two sets to the size of each set. The similarity between two documents which is computed from any of these three equations is normalized to a value between zero and one. If two documents are encoded into identical bags of words, their similarity is 1.0, while if they are encoded into two bags of words with no shared word, their similarity is 0.0.

$$\textbf{Dice Index } sim(d_i, d_j) \approx sim(d_i^b, d_j^b) = \frac{2\left|d_i^b \cap d_j^b\right|}{\left|d_i^b\right| + \left|d_j^b\right|} \quad (1)$$

$$\textbf{Jaccobian Index } sim(d_i, d_j) \approx sim(d_i^b, d_j^b) = \frac{\left|d_i^b \cap d_j^b\right|}{\left|d_i^b \cup d_j^b\right|} \quad (2)$$

$$\textbf{Cosine Index } sim(d_i, d_j) \approx sim(d_i^b, d_j^b) = \frac{1}{2}\left(\frac{\left|d_i^b \cap d_j^b\right|}{\left|d_i^b\right|} + \frac{\left|d_i^b \cap d_j^b\right|}{\left|d_j^b\right|} \right) \quad (3)$$

In the three equations, a similarity between two bags of words encoding documents is computed based on the portion of their shared words, but these three equations generate different output values. For example, two documents are encoded into a bag of 200 words and a bag of 300 words. In this example, both bags share 50 words among words contained in them. Values of $\left|d_i^b\right|$ and $\left|d_j^b\right|$ are 200 and 300, respectively, and values of $\left|d_i^b \cap d_j^b\right|$ and $\left|d_i^b \cup d_j^b\right|$ are 50 and 450. Therefore, equation (1), (2), and (3) generate 0.2, 0.1111, and 0.2083, respectively by substituting these values for the equations.

2.1.2 A Numerical Vector

This subsection describes the traditional way to encode documents. In this method, documents are represented into numerical vectors. The main machine learning algorithms such as NB (Naïve Bayes), SVM (Support Vector Machine), MLP (Multi-Layer Perceptron), and Kohonen Networks use numerical vectors as their input vectors. This method is necessary to apply such machine learning algorithms for text categorization and text clustering. This subsection describes such a way briefly and points out its disadvantages.

The features of the numerical vectors representing documents are frequent words occurring in the document collection. If all the words were used as features, the dimension would be more than 10,000; such a dimension is not feasible for text categorization and text clustering. Previous research suggested several methods for feature selection, such as mutual information, chi-square, frequency based method, and information theory based methods [Sebastiani 2002]. The words, which are selected as the features of the numerical vectors, are denoted by $w_1, w_2, ..., w_n$.

There are three common ways to define feature values in numerical vectors representing documents. The first way is to use a binary value, zero or one, for each word; zero indicates its absence and one indicates its presence in the given document. The second way is to use the frequency of each word in each document as its value. In this way, each element is an integer. The third way is to use the weight of each word, w_k $(1 \le k \le n)$, as a feature value based on its term frequency and its inverse document frequency. Such a weight is computed from equation (4),

$$weight_i(w_k) = tf_i(w_k)(\log_2 D - \log_2 df(w_k) + 1) \quad (4)$$

where $tf_i(w_k)$ is the frequency of w_k in the document, d_i, D is the total number of documents in a given corpus, and $df(w_k)$ is the number of documents including the word, w_k, in the corpus. The feature value corresponding to each of the selected words, w_k is expressed as a function of the word, $\phi(w_k)$, where its output is a binary value indicating its presence, an integer value indicating its frequency, or a real value indicating its weight computed from equation (4).

Two numerical vectors representing two documents, d_i and d_j, are denoted by $d_i^f = [\phi_i(w_1) \quad \phi_i(w_2) \quad .... \quad \phi_i(w_n)]$ and $d_j^f = [\phi_j(w_1) \quad \phi_j(w_2) \quad .... \quad \phi_j(w_n)]$, where its

dimension is n. The content similarity between two documents, d_i and d_j, denoted by $sim(d_i, d_j)$, is computed using one of Dice Index, Jaccard Index, and Cosine Index which are expressed with equation (5), (6), and (7), respectively. These three equations are used commonly and traditionally in clustering problems [Duda, et al. 2001], once examples are represented as numerical vectors. In all of equation (5), (6), and (7) [Salton 1989], the computation is based on the ratio of the inner product of the two numerical vectors to the addition of their norms.

$$\text{Dice Index}: sim(d_i, d_j) \approx sim(d_i^n, d_j^n) = \frac{2(d_i^f \cdot d_j^f)}{\left\|d_i^f\right\|^2 + \left\|d_j^f\right\|^2} \quad (5)$$

$$\text{Jaccard Index}: sim(d_i, d_j) \approx sim(d_i^n, d_j^n) = \frac{(d_i^f \cdot d_j^f)}{\left\|d_i^f\right\|^2 + \left\|d_j^f\right\|^2 - d_i^f \cdot d_j^f} \quad (6)$$

$$\text{Cosine Index}: sim(d_i, d_j) \approx sim(d_i^n, d_j^n) = \frac{(d_i^f \cdot d_j^f)}{\sqrt{\left\|d_i^f\right\|^2 + \left\|d_j^f\right\|^2}} \quad (7)$$

For example, two documents are represented into two four dimensional numerical vectors, [1 1 0 1] and [1 0 1 0]. Values of $\left|d_i^f\right|$ and $\left|d_j^f\right|$ are 3 and 2. The inner product of the two vectors, $\left(d_i^f \cdot d_j^f\right)$, becomes 1. Therefore, equation (5), (6), and (7) generate 0.4, 0.25, and 0.4472, respectively by substituting these values for the equations.

This way of representing documents causes two main problems, as discussed in section 1.2: the huge dimensionality and the sparse distribution of numerical vectors. The first problem, the huge dimensionality, leads to very high computation cost for document processing. Although feature selection methods are used as solutions to this problem, dimensions of numerical vectors representing documents remain large. The second problem, the sparse distribution of numerical vectors, leads to poor discrimination among numerical vectors, since almost all the numerical vectors representing documents are close to empty vectors (vectors with zero values). If empty vectors are present across several categories in text categorization, the performance is very much degraded.

2.2 Text Clustering

In the previous section, we described the two existing representations of documents as preprocessing for text clustering or text categorization, along with computation of their association similarities. In this section we will explore previous research on text clustering in the first subsection, and describes the three main existing approaches to text clustering: single pass algorithm, Kohonen Networks, and EM algorithm in the second subsection.

2.2.1 Previous Works on Text Clustering

In this subsection we will explore previous research on text clustering and its relevant topics. At first, an incremental clustering system, COBWEB, will be introduced as the basis for the idea of this research, and its limits and differences from this research will be described. Second, existing text clustering systems will be surveyed and their limitations

will be discussed. Finally, the EM algorithm will be introduced as a framework for clustering objects, and previous applications of EM to text clustering and its various versions will be presented.

In 1987, Fisher proposed an incremental concept clustering system, called COBWEB [Fisher 1987]. This system could be the basis for a DDO system, in that the incorporation of object clustering into object classification was attempted. However, as discussed below, it was targeted for generic structured data, instead of textual data. In COBWEB, the operation of building and maintaining the classification tree consists of classification, new class creation, class merging, and class splitting. Among them, three operators, new class creation, class merging, and class splitting correspond to the creation mode in DDO system, where documents are organized initially or again. The operation, classification, corresponds to our DDO system's maintenance mode, where successive documents are added to the organization. COBWEB optimizes its classification tree by applying one of the operators toward maximization of its category utility which is an evaluation measure for the system based on Bayes rule. In other words, the system builds and evolves its classification tree using a strategy of hill- climbing in terms of its category utility.

However, COBWEB, has serious limitations when implementing Dynamic Document Organization tasks. Basically, it was designed to cluster simple generic structured data in which each attribute has discrete values, rather than textual data; it was not intended for text clustering. If this system was applied to numerical vectors with continuous values representing objects, these values would have to be discretized. Furthermore, since the system optimizes its classification tree using a strategy of hill climbing with respect to its category utility, it may fall into local optima, and, thus, fail to find the global optimum. Although COBWEB could provide a basis for implementing DDO systems as an idea, it is not suitable for the DDO tasks in its original form.

Kaski and his colleagues developed WEBSOM, the system of document clustering with SOM (Self Organizing Map)[1] in 1998 [Kaski, et al. 1998]. WEBSOM displays the results of text clustering graphically for browsing a collection of documents [Kaski, et al. 1998]. In WEBSOM, a word category map is built in the competitive layer; each node in the layer corresponds to a set of similar words. Like for text categorization, each document was represented as a feature vector; its dimension was 315 [Kaski, et al. 1998]. This system performs text clustering by modifying the weights between the feature vector and the winner node in competitive layer. The winner node corresponds to a set of similar words nearest to feature vector in Euclidean distance. Kohonen Networks will be described more detail in subsection 2.2.2.2.

T. Kohonen and his colleagues implemented a modified version of WEBSOM to apply it, more practically, to a massive document collection consisting of 6,840,567 patent abstracts, in 2000 [Kohonen, et al. 2000]. This modified version was identical to the

[1] SOM was proposed by Kohonen and also called Kohonen Networks [Kohonen 1982]. It uses numerical vectors as its input vectors and its weight vectors. Its input is a collection of unlabeled numerical vectors and its output is a list of clusters and optimized weight vectors corresponding to them. The number of clusters is given as the parameter of the Kohonen Networks, and weight vectors as the prototypes of clusters are initialized with random values. For each input vector and each prototype, the nearest weight vector in Euclidean distance or cosine similarity is updated toward inter values between these two vectors. Finally, weight vectors are optimized by repeating these updates. Since Kohonen Networks organized unlabeled input vectors by itself, it is called SOM.

previous version [Kaski, et al. 2000] with respect to the process of text clustering. In the second version, only the definition of the data structure for text clustering was modified in order to improve the speed of text clustering using WEBSOM, although they used Kohonen Networks as the approach. The modified version was compared to the previous version on 6,840,567 patent abstracts with respect to their accuracy and speed. The result of that research showed that the modified version reduces time for text clustering by 90% while maintaining its performance [Kohonen, et al. 2000].

In 2000, Hatzivassiloglou, et al claimed that there are two types of clustering algorithms: *hierarchical clustering algorithms* and *nonhierarchical ones* [Hatzivassiloglou, et al. 2000]. The former type is the class of clustering algorithms in which critical similarity is given as a parameter and is the critical value for determining whether a document belongs to one of the existing clusters or whether a new cluster need to be recreated. This class of clustering algorithms includes single link, complete link, group-wise average, and single pass. The latter type is the class of clustering algorithms in which the number of clusters is given as a parameter, selects representative vectors for each cluster, and populates clusters, based on the similarity between the input vector and the representative vector of each cluster. Such types of clustering algorithms include the k-means algorithm and Kohonen Networks. The number of clusters in the first type of clustering algorithms is set dynamically depending on the distribution of the document similarities, while the number of clusters in the second type is given as a parameter, without any regards for such a distribution. Therefore, hierarchical clustering algorithms are more natural and practical than nonhierarchical ones, at this point.

Among hierarchical clustering algorithms, the single pass algorithm is the most practical one because of its fast computation and its simplicity. As previously discussed its parameter is the *critical similarity* which decides whether a document should be included in an existing cluster or in a new one. Initially, a cluster is created and the first document is included in that cluster. For successive documents, the similarity between a successive document and the existing clusters is computed. Since each cluster may have more than two documents, the similarity between a document and a cluster is the average similarity of the document with the documents inserted in the cluster. The maximum similarity of the document is selected. If the maximum similarity is larger than the critical similarity, the document is inserted in the existing cluster corresponding to it. Otherwise, a new cluster is created and the document is included in it. This process continues until no more documents need to be clustered. This algorithm will be described in more detail in subsection 2.2.2.

Bote and his colleagues developed a text clustering system using Kohonen Networks in 2002 [Bote, et al. 2002]. Their proposed system of text clustering is identical to WEBSOM in that Kohonen Networks is used as the algorithm for text clustering, but their system presents some differences with WEBSOM. First, WEBSOM builds word category maps in the competitive layer, while their system of text clustering builds 20*20 unnamed nodes in the layer. Second, WEBSOM targeted unlabeled documents [Kohonen, et al. 2000][Kaski, et al. 2000], while this system targets labeled ones [Bote, et al. 2002]. The goal of their research is not to cluster documents but to map labeled documents into 20 * 20 nodes, topologically. Hence the two previous studies are identical with respect to text clustering using Kohonen Networks, but they are different with respect to their goal.

In 1977, Dempster and his colleagues proposed the EM algorithm, initially, as an iterative algorithm for estimating maximum likelihood of incomplete data [Dempster, et al 1977]. Afterward, EM algorithm has been used as a clustering algorithm for generic structured data [Celeux and Govaert 1992] [Ambroise and Govaert 1998] and as an approach to text clustering. In 2000, Vinokourov and Girolami proposed five probabilistic models of hierarchical text clustering based on the frame of the EM algorithm [Vinokourov and Girolami 2000]. In their study, each of the proposed five probabilistic approaches to text clustering consists of *E-step (Expectation step)* and *M-step (Maximization step)* which are the main and the basic procedures of the EM algorithm. The five probabilistic models were defined depending on how to perform the specific computations involved in the E-step and the M-step and which distribution is assumed in each cluster. 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 [Banerjee, el al 2003]. These two studies show that any type of raw data should be represented as numerical vectors, in order to use the EM algorithm. The EM algorithm is applicable only to numerical vectors, because the operations involved in its two steps are only possible for numeric data.

In the following sub-sections we will describe in greater detail three of the existing unsupervised learning algorithms commonly used for text clustering. We describe their process, their application to text clustering, and their properties. The first subsection will describe the single pass algorithm which is a simple and fast clustering algorithm and is recommended for implementing real time document clustering systems. The second section will cover Kohonen Networks which is a typical and popular neural network for unsupervised learning and performs its task by competition of its nodes. WEBSOM is a typical document clustering system which adopts Kohonen Networks in the approach [Kohonen, et al. 2000][Kaski, et al. 2000]. The last subsection will cover the EM algorithm which consists of two main steps, the *E-step* and the *M-step*, and performs its unsupervised learning by repeating the two steps until the parameters of the clusters converge to their optimized values.

2.2.2 Single Pass Algorithm

The single pass algorithm is a data clustering algorithm based on the comparison of the maximum similarity between a particular data item and existing clusters with the critical similarity, parameter. Other data clustering algorithms, such as Kohonen Networks and the k-means algorithm, use the number of clusters as their parameter. This approach determines the number of clusters by itself, depending on the value of the critical similarity. The higher the critical similarity, the more clusters of small size are created. The lower the critical similarity, the fewer clusters of large size are created. Since the similarity between documents is a normalized value as discussed in section 2.1, the critical similarity is given as a normalized value as well.

Figure 4 illustrates the process of clustering documents using the single pass algorithm. A list of documents and the critical similarity are given as the input to this process. These documents are encoded into bags of words, or numerical vectors. The process of encoding them was already described in section 2.1. Initially, a cluster is created and the first document is included in it, as its representative document which is used to compute

the similarity between its cluster and another document. For each successive document, sub-steps, 4-1, 4-2, and 4-3, in figure 4, are repeated. The similarities of the representative documents corresponding to existing clusters with each successive document are computed using one of the strategies described in section 2.1. The representative document of each cluster indicates the document which is included initially, when the cluster is created. The maximum of these similarities and its corresponding cluster are determined. If the largest similarity is higher than or equal to the critical similarity, the successive document is included in its corresponding cluster. Otherwise, a new cluster is created and the successive document is included in the new cluster as its representative document. The process illustrated in figure 4, thus, generates a list of clusters including documents, as its output.

Input: a list of documents and the critical similarity
Step 1: Encode a list of documents into bags of words, numerical vectors, or string vectors
Step 2: Get the first document from the list of documents
Step 3: Create a cluster and include the document into it, as its representative document
Step 4: Repeat the following substeps, 4-1, 4-2, and 4-3 for each of successive documents
Step 4-1: C ompute the similarity between the successive document and the representative document of each cluster
Step 4-2: Obtain the maximum similarity and its corresponding cluster among existing clusters
Step 4-3: If the maximum similarity >= the critical similarity, include the successive representation otherwise, go to step 3
Output: a list of clusters including documents

Figure 4. The process of clustering documents with single pass algorithm

An advantage of this approach is that it is applicable to any strategy for encoding documents, once the method for computing a similarity between encoded documents is established. Since the similarity between documents is computed as different values depending on the strategy for encoding documents, this approach can result in different clusters of documents.

Another advantage is that the process of clustering documents using this approach is performed almost linearly in the number of documents, if the number of clusters is far less than the number of documents. Let's assume that there are n documents for clustering and m clusters. Each document is represented into a d dimensional numerical vector. If the initial document of each cluster is compared with successive documents as its prototype, its complexity is $O(dmn)$; it is almost linear to the number of documents, if the number of clusters is far less than the number of documents ($m << n$). In this approach, text clustering is performed with almost linear complexity on the number of documents, when one thousand documents are given as its input, and the final number of output clusters is less than five.

A disadvantage of the single pass algorithm is that the cluster prototypes are not optimized during its computation. The Kohonen Network algorithm updates the prototypes continuously, during the learning process. The single pass algorithm uses the first document of each cluster as its prototype. This means that the result of text clustering depends on the order in which documents are processed. This is the price to pay for its fast computation.

2.2.3 Kohonen Networks

In this study, the second approach to text clustering is Kohonen Networks. It was proposed by Kohonen in 1982 initially as a neural network for unsupervised learning [Kohonen 1982]. But after eighteen years, in 2000, it started to be applied to text clustering [Kohonen, et al. 2000]. Between 1982 and 2000, it had been applied to other tasks, such as forecasting heavy load on electrical power system [Baumann, et al. 1993], driving vehicle automatically with vision system [Malmstrom, et al. 1994], solving the traveling salesman problem [Aras, et al. 1999], and clustering emission signals [Emamian, et al. 2000]. Similarly to MLP (Multi Layers Perceptron) Back Propagation, which is the most popular neural network for supervised learning, Kohonen Networks is the most popular neural network for unsupervised learning [Ritter, et al. 1991].

Figure 5 shows the architecture of Kohonen Networks which consists of the input layer and the competitive layer. There are complete connections between these two layers, and each connection corresponds to a weight which indicates an element of the weight vectors as the prototypes for the given clusters. The conditions concerning the nodes and the weights of these two layers are listed as follows.

- *The number of nodes in the competitive layer is identical to the number of clusters given as a parameter.*
- *The number of nodes in the input layer is identical to the dimension of the input vectors encoding documents.*
- *Each node in the competitive layer corresponds to each weight vector.*
- *The number of weight vectors is identical to the number of nodes in the competitive layer, and their dimension is identical to that of the input vectors.*

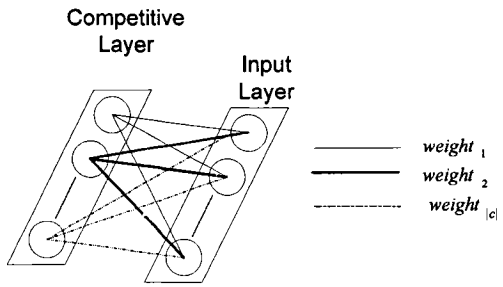

Figure 5. The architecture of Kohonen Networks and NTSO

Figure 6 illustrates the process of clustering documents using Kohonen Networks. At first, a list of documents is given as the input, and the number of clusters, the number of iterations, and learning rate, denoted by η, are given as the parameters. These documents are encoded into numerical vectors clustering the preprocessing phase. The architecture of Kohonen Networks is built, following the above conditions. The elements of each weight vector are initialized differently and randomly. The similarity between each input vector encoding a document and each weight vector indicating the prototype of a cluster is computed using equations (5), (6), or (7). This similarity is the value of each competitive node to the given input vector. Only the weight vector of the competitive

node with the highest similarity value[2] is updated for each input vector, using equation (8),

$$weight_{\max}(t) = weight_{\max}(t-1) + \eta(input_h - weight_{\max}(t-1)) \quad (8)$$

where $weight_{\max}$ is the weight vector, η is the learning rate given as a parameter, and $input_h$ is a particular input vector encoding a document. The process of computing the values of the competitive nodes and updating the weight vector for each input vector is repeated as many times as the number of iterations given as a parameter. After this iteration, for each input vector, the values of competitive nodes are computed using the optimized weight vectors. Each document is arranged into the cluster corresponding to the competitive node with the highest value; it is included in the cluster with the most similar prototype as its corresponding input vector. Finally, this process generates a list of clusters including documents as its output, as in the process illustrated in figure 4.

As discussed above, Kohonen Networks have been applied not only to text clustering but also to other tasks [Baumann, et al. 1993][Malmstrom, et al. 1994][Aras, et al. 1999][Emamian, et al. 2000]. Its advantage over the single pass algorithm is that the prototypes of clusters are optimized through its unsupervised learning. These optimized prototypes are expected to lead to clusters with improved intra-cluster similarities over those greatest by the simple pair algorithm. But its disadvantage is that it is a more expensive algorithm for text clustering than the single pass algorithm. In particular the complexities of clustering documents with Kohonen Networks and single pass algorithm are defined as follows.

- **Kohonen Networks**: *the number of documents * the number of iterations * the number of clusters*
- **Single pass algorithm**: *the number of documents * the number of clusters*

In this approach, another disadvantage is that the given documents should be encoded into numerical vectors in spite of the risk of the two main problems discussed in section 1.2.

Input and parameters: a list of documents, #clusters, #iteration, and learning rate
Step 1: Encode a list of documents into numerical vectors as input vectors
Step 2: Set up the architecture of Kohonen Networks 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 with random values
Step 4: repeat the 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
Step 4-1-2: update the weight vector with the highest similarity with equation (8)
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 6. The process of clustering documents using Kohonen Networks

Let's assume that there are n documents and m clusters, and the Kohonen Networks is trained with k iterations. Each document is represented into a d dimensional numerical

[2] Since the value of each competitive node is normalized, the higher the value, the closer to one.

vector. Its complexity of clustering documents, using the Kohonen Networks becomes $O(dkmn)$. For each document, it costs m running times for selecting a winning node. It costs mn running times per iteration for training the Kohonen Networks. Therefore, totally, it costs kmn running times for training it with k iterations.

2.2.4 EM Algorithm

This subsection describes the EM algorithm which is another popular unsupervised learning algorithm. We describe the learning algorithm, its application to text clustering, and its properties. The EM algorithm was initially proposed by Dempster and his colleagues in 1977 as an iterative algorithm for estimating maximum likelihoods of incomplete data [Dempster, et al 1977], as mentioned in section 2.2.1 and consists of two main steps as follows.

- **E-Step (Expectation)**: *computes the probabilities that each object belongs to clusters. This corresponds to their likelihoods based on the current parameters characterizing the clusters*
- **M-Step (Maximization)**: *updates the parameters of the clusters by replacing the current parameters by new parameters based on the probabilities computed from the E-step*

It performs unsupervised learning by repeating the above two steps until parameters of clusters converge to their optimized values. This version of the EM algorithm makes the following two assumptions:

- *Each object is represented as a numerical vector.*
- *Numerical vectors representing objects in each cluster follow multivariate Gaussian distribution whose parameters are mean vector and covariance matrix.*

We try to cluster n examples into c clusters using EM algorithm. Before the first E-step, parameters characterizing clusters should be initialized. The number of clusters should be determined initially as an input parameter and initial clusters are built at random. Then, mean vectors and covariance matrices can be easily computed as initial parameters for entering the E-step from the numerical vectors in clusters.

In the E-step, the probabilities that each object belongs to clusters are computed with the likelihood. Let $P(x_i \mid \theta_j)$ be the probability that a numerical vector, $x_i, 1 \le i \le n$ representing the i th object belongs to the j th cluster characterized by the parameters θ_j $1 \le j \le c$. Since it is assumed that all the numerical vectors representing objects in each cluster follow an arbitrary multivariate Gaussian distribution, the parameters characterizing the j th cluster are the mean vector, μ_j, and its covariance matrix, Σ_j. The probability, $P(x_i \mid \theta_j)$, given as the likelihood of each object to the jth cluster, is computed, using equation (9), based on the multivariate Gaussian distribution,

$$p(x_i \mid \theta_j) = \frac{\exp(-\frac{1}{2}(x_i - \mu_j)^t \Sigma_j^{-1}(x_i - \mu_j))}{\sum_{k=1}^{m} \exp(-\frac{1}{2}(x_i - \mu_k)^t \Sigma_k^{-1}(x_i - \mu_k))} \quad (9)$$

where m indicates the number of clusters determined as an input parameter in the step of initialization. Thus, equation (9) indicates that an estimated value of $P(x_i \mid \theta_j)$ is the rate of probability density functions in all clusters.

Once the probabilities that each object belongs to clusters are computed using equation (9), this algorithm enters the M-step where the new parameters of the clusters are computed with respect to the maximum likelihood of objects with respect to the clusters. The parameters are then used to replace the previous ones. In order to compute the new parameters, equation (10) which computes a sample mean vector and equation (11) which computes a sample covariance matrix are derived [Mitchell 1997] [Duda el al 2001]. The new mean vector, μ_j' and new covariance matrix, Σ_j' of the j th cluster are computed, using these equations

$$\mu_j' = \frac{1}{n}\sum_{i=1}^{n} p(x_i \mid \theta_j) x_i \quad (10)$$

$$\Sigma_j' = \frac{1}{n}\sum_{i=1}^{n} p(x_i \mid \theta_j)(x_i - \mu_j')^t (x_i - \mu_j') \quad (11)$$

where n indicates the total number of objects. If there is little difference between the previous parameters and the new ones, this algorithm terminates. Otherwise, it enters the E-step again. It will repeat the two steps until these parameters converge to their optimized values.

The EM algorithm, itself, produces, upon convergence, optimized values of probabilities and parameters of clusters as its output. If this algorithm is used for hard clustering, where each object is allowed to belong to only one cluster, each object is placed in the cluster corresponding to the highest, $\arg\max_j P(x_i \mid \theta_j)$ value. Otherwise, several heuristic methods may be considered.

Figure 7 illustrates the process of clustering documents using the EM algorithm. The input consists of a list of documents, the number of clusters, and the number of iterations for process. All the steps of the algorithm were previously discussed.

Input and parameters: a list of documents, #clusters, and #iteration
Step 1: Encode a list of documents into numerical vectors as input vectors
Step 2: build initial clusters using random clustering
Step 3: compute initial parameters, mean vector and covariance matrix, from each initial cluster
Step 4: repeat the step 4-1 #iteration times
Step 4-1: do the step 4-1-1 to each numerical vector encoding document i
Step 4-1-1: do the step 4-1-1-1 to each cluster j
Step 4-1-1-1: compute probability that document i belongs to cluster j using equation (9)
Step 4-2: do the steps 4-2-1 and 4-2-2 to each cluster j
Step 4-2-1: compute the new mean vector and the covariance matrix using equation (10) and (11)
Step 4-2-2: replace previous mean vector and covariance matrix by new ones from 4-2-1
Step 5: arrange each document into the cluster corresponding to its highest probability
Output: a list of clusters including documents

Figure 7. The Process of Clustering Documents using EM Algorithm

Since the EM algorithm is able to search for optimized parameters of the clusters, even if the initial parameters are set arbitrary, it has been used as one of state of the art clustering algorithms [Celeux and Govaert 1992] [Ambroise and Govaert 1998]

[Vinokourov and Girolami 2000] [Banerjee el al 2003]. A disadvantage of this algorithm however is that like the gradient descent method in back propagation, this algorithm may fall into one of the local optima, instead of the global optimum. Another disadvantage is that this version of the algorithm requires encoding of documents into numerical vectors. This, as discussed previously, causes huge dimensionality and sparse distribution. If all the numerical vectors representing documents have very sparse distributions, neither the EM algorithm nor any other clustering algorithm can be expected to work well for text clustering.

Let's assume that there are n documents and m clusters, and E-step and M-step is repeated k times. Each document is represented into a d dimensional numerical vector. Its complexity of clustering documents, using the EM algorithm becomes $O(dkmn)$. For each document, it costs dm running times for computing probabilities that it belongs to clusters in the E-step. It costs dmn running times in the entire E-step. In the M-step, it costs n running times for compute parameters satisfying the maximum likelihood for each cluster; in the entire M-step, it costs also dmn running times. If the E-step and M-step are repeated alternatively k times, its running time becomes $2dkmn$. Therefore, the complexity of EM-algorithm for clustering documents becomes $O(dkmn)$.

2.3 Previous Research on Cluster Identification

Because text clustering produces unnamed clusters of documents, each cluster needs to be labeled, through cluster identification, in order to access a document by browsing. In the previous chapter, the three conditions for cluster identification were already defined. Previous research on text clustering also considered how to label clusters. Both versions of WEBSOM labeled clusters with a word category map in the competitive layer of SOM; each node in such layer corresponds to a set of semantically similar words [Kohonen, et al. 2000][Kaski, et al. 2000]. The goal of labeling clusters in WEBSOM is to display a semantic map of the document collection visually; its goal differs from the goal of this research with respect to cluster identification. It is very expensive to build such a word category map; it requires creating a cluster of words in advance by representing each word into a feature vector. But previous research on WEBSOM applied text clustering only once; they assumed that the organization of documents was permanently fixed once it is built through text clustering; this is not practical in the real world.

Wu and his colleagues proposed a clustering based interface and a categorization based one for interactive information retrieval in 2001 as alternatives to list based interfaces [Wu, et al. 2001]. A clustering based interface is the interface displaying retrieved documents, not sequentially but with clusters. Their research used cluster descriptors to identify clusters of retrieved documents. Each cluster descriptor consists of eighteen words: the ten words with highest weights from full texts, the three words from titles, and the five word pairs with highest frequencies. Their research is also different from this research in its goal. The goal of their research is for users of information retrieval systems to word immediately the content of each cluster, while the goal of this research is the preparation of labeled sample documents for text categorization as well as the identification of clusters. Therefore, their cluster descriptor is too long to be used as the labels of documents, although their approach satisfies two of our three conditions for cluster identification.

2.4 Text Categorization

In this section, we will survey previous research on text categorization and describe the main traditional approaches, since the implementation of DDO involves it. First, rule based approaches and machine learning based approaches to text categorization will be reviewed and the advantages of the latter type over the former type will be described. Second, we will present the basic principles of machine learning based approaches and show that previous research follows those principles. Third, we will survey in detail previous cases of applying Naïve Bayes, SVM, and Back Propagation, to text categorization, to justify the adoption of these approaches for comparison with ours. Fourth, we will survey previous solutions that use both labeled and unlabeled sample documents for text categorization, since labeled documents are expensive while unlabeled ones are cheap. Note that text categorization covers just a small fraction of the tasks involved in DDO.

There are two types of approaches to text categorization: rule based and machine learning algorithm based [Sebastiani 2002]. In the former type of approaches, classification rules are built manually and unseen documents are classified based on these rules. In the latter type, such rules are built automatically by analyzing a sample of labeled documents statistically. Rule based approaches have high *precision*[3] but very low *recall*[4] because of the lack of flexibility, while machine learning based approaches show a greater balance between precision and recall. This is because of their greater flexibility, although such approaches have lower precision than rule based approaches. If the domain of documents targeted for text categorization is highly specialized (e.g. engineering or medicine), rule based approaches are useless without experts in such domain, because they require prior knowledge to build the rules. Therefore machine learning based approaches are generally preferred to rule based approaches [Sebastiani 2002].

Since text categorization is a popular research area, there has been a lot of previous research on it. Sebastiani surveyed ten-years of previous research on text categorization in 2002 [Sebastiani 2002]. He insisted that text categorization is a very important task especially for information retrieval, and recommends machine learning based approaches, rather than rule based approaches [Sebastiani 2002]. His article [Sebastiani 2002] mentioned the application of text categorization to document organization briefly, presented more than ten machine learning based approaches, and stated that there are more approaches in addition to those. His survey is very significant for this research as a tutorial on text categorization.

2.4.1 Previous Research on Traditional Approaches: NB, SVM, and BP

This study adopts NB (Naïve Bayes), SVM (Support Vector Machine), and BP (Back Propagation), as traditional machine learning based approaches for comparison with the techniques we will describe in chapter 4. Note that these three traditional approaches require the representation of documents into numerical vectors, nonetheless they have

[3] Precision refers to the rate of the correctly classified positive documents to all of the classified positive documents.

[4] Recall refers to the rate of the correctly classified positive documents to all of the true positive documents.

their own advantages[5] [Sebastiani 2002]. Among them, NB is the most popular and traditional approach [Mitchell 1997]. It classifies each instance using a specialized version of the Bayes rule, under the assumption that the features of each numerical vector are independent from each other. Classification using a generic Bayes rule consists of computing the posteriori probability of each class given the instance and assigning the class with maximum value to the instance. Because of the independence assumption in Naive Bayes, the posteriori probability of each class given a numerical vector representing an instance is expressed as the product of posteriori probabilities given in that vector.

SVM is the second most popular approach. It works better for text categorization on the standard test bed, Reuter 21568, than any other machine learning based approaches [Sebastiani 2002]. It defines various types of kernel functions to map training examples in a nonlinearly separable space into those in a linearly separable space [Platt 1998]. The outputs from its training examples are two hyper-planes corresponding to the boundaries of the two classes, in the linearly separable space with the maximum margin; it was designed for binary classification, originally, although there are now many variations for multiple- classification problems [Rennie 2001].

There are two solutions for training examples in nonlinearly separable spaces where a linear classifier, such as the Perceptron[6], is not applicable. The first solution is to map the training examples of that space into a linearly separable space using a kernel function. This is the method adopted in SVM. The second solution is to define nonlinear boundaries between classes by adding one more layer, called a hidden layer, between the input and the output layer of a Perceptron, and that is the solution adopted in BP. Therefore, both SVM and BP are derived from Perceptrons and designed to address the fact, that they are not applicable to training examples in nonlinearly separable spaces. In the BP learning algorithm, the weights between layers are initialized with random values around zero. Output values are computed with the current values of weights, in the forward direction, from input layer to output layer. Based on the error between the computed output values and the targeted output values, the weights are updated in the backward direction, from output layer to input layer. The advantages of BP are that it could be applied not only to pattern classification but also to nonlinear regression [Haykin 1994], and that it is tolerant to noise in training examples [Mitchell 1997][Haykin 1994]. NB, SVM, and BP will be described in depth in section 2.4.3.

In the meantime, we will present previous research that uses one of these three approaches to text categorization in order to justify their adoption for comparison. First, NB (Naïve Bayes) is the most popular approach to text categorization among them. Mitchell presented the application of NB (Naïve Bayes) to text categorization in his text book in 1997 [Mitchell 1997]. Actually, the attributes of the feature vectors representing documents are not independent in text categorization, and this violates the assumption of NB. He insisted that Naïve Bayes worked well in text categorization, in spite of that violation [Mitchell 1997]. Mladenic and Grobellink evaluated feature selection methods

[5] NB has the fastest learning process among the three approaches, SVM results in the best performance in text categorization [Sebastiani 2002], and BP is applicable to both pattern classification and nonlinear regression.

[6] Perceptron is a simple neural network that consists of input layer and output layer. It is a linear classifier, which is applicable only to a linearly separable space in pattern classification.

for the application of Naïve Bayes to text categorization under an imbalanced distribution over the classes in 1999 [Mladenic and Grobelink 1999]. They recommended the odd's ratio as the optimal feature selection method under an imbalanced distribution. It is explained in detail in the literature, [Mladenic and Grobelink 1999]. Their research implied that Naïve Bayes is a very popular approach to text categorization [Mladenic and Grobelink 1999]. Androutsopoulos and his colleagues applied Naïve Bayes to spam mail filtering, which is a very important task for users of the Internet and a very practical instance of text categorization [Androutsopoulos, et al. 2000].

Previous research on the application of SVM to text categorization will be surveyed to show that it is also popular, as an approach to text categorization. Joachims is the first researcher to have proposed SVM (Support Vector Machine) as an approach to text categorization [Joachims 1998]. He compared SVM with Naïve Bayes and k Nearest Neighbor in text categorization on Reuter 21578, a standard test bed, and empirically showed that SVM is the best of three approaches for that problem [Joachims 1998]. Cristianini and Shawe-Taylor presented a case of applying SVM to text categorization in their text book, in 2000 [Cristianini and Shawe-Taylor 2000]. H. Drucker, Donghui Wu, and V. N. Vapnik applied SVM to spam mail detection, which is a practical instance of text categorization, in 1999 [Drucker, et al. 1999]. They argued that SVM is a better approach for that problem than Naïve Bayes. Sebastiani surveyed previous research on text categorization and showed experimentally that SVM is the best approach to text categorization on Reuter 21578 [Sebastiani 2002].

BP (Back Propagation) is the most popular model among neural networks for any problem, although it is not as popular as NB (Naïve Bayes) and SVM (Support Vector Machine), for text categorization. Wiener applied BP (Back Propagation) to text categorization on Reuter 21578 in his thesis, and showed empirically that BP (Back Propagation) is better than k Nearest Neighbor [Wiener 1995]. Ruiz and Srinivasan proposed HME (Hierarchical Mixture of Experts), which is a combined model of a BP, as a text categorizer, and showed that HME is better than a flat mixture in text categorization [Ruiz and Srinivasan 2002].

The application of traditional machine learning algorithms such as NB (Naïve Bayes), SVM (Support Vector Machine), and BP (Back Propagation), to text categorization requires the representation of documents into numerical feature vectors. Such a representation causes two problems: a huge dimensionality and a sparse distribution of feature vectors. The first problem is that many words are required to represent documents as attributes of feature vectors and that their dimension can reach ten of thousands. The second problem is that each feature vector has zero values in more than ninety percent of their features in spite of the presence of many attributes. Previous research proposed feature selection methods to address these problems [Sebastiani 2002][Cohen 1995][Joachims 1998]. Feature selection did address these problems by reducing the dimension of the feature vectors by selecting the most relevant attributes. In these methods, however excessive reduction of feature vectors size causes very high information loss, damaging classification performance. Feature selection methods, thus have limits in addressing such problems. This research will address these problems by representing documents into string vectors instead of feature vectors.

2.4.2 Previous Research on using Labeled and Unlabeled Documents

Text categorization requires two preliminary tasks: the predefinition of categories and the manual labeling of sample documents, before learning them and classifying unseen documents. Manual labeling of documents consists of two steps for each document: reading it and deciding on its category; it takes very much time to do that for a large collection of sample documents. If the preliminary task costs that much time, text categorization itself is very expensive and impractical for information systems. When the classification predefinition of an information system is changed, the sample documents should be relabeled to reflect the updated definition. Note that just the manual labeling of sample documents is very expensive.

Previous research proposed solutions for the situation where labeled documents are very expensive, while unlabeled ones are cheap [Nigam, et al. 2000][Skarmata, et al. 2000]. Such solutions are characterized by the utilization of both labeled and unlabeled documents as sample documents. Nigam and his colleagues proposed the combination of EM (Expectation Maximization) and NB (Naïve Bayes) to use both kinds of sample documents in 2000 [Nigam, et al. 2000]. Naïve Bayes is trained with a labeled sample of documents initially as a text classifier. It assigns a probability of appearance in each class to the unlabeled sample documents. It is then retrained with the originally labeled and the unlabeled documents categorized with their projected class. It reassigns class appearance probabilities to the initially unlabeled documents. This process is repeated until the probabilities assigned to the initially unlabeled documents converge.

Skarmeta and his colleagues proposed an alternative solution to the combination of NB (Naïve Bayes) and EM (Expectation Maximization) in 2000 [Skarmata, et al. 2000]. This solution is the combination of ssAHC (Semi-Supervised Agglomerative Hierarchical Clustering) and k Nearest Neighbor to use labeled and unlabeled sample documents. Before k Nearest Neighbor performs the classification, unlabeled examples are clustered by ssAHC based on the classification system of the labeled examples. Skarmeta and his colleagues compared their proposed solution with the combination of Naïve Bayes and EM and showed empirically that their proposed approach works better than the basic combination of Naïve Bayes and EM in text categorization [Skarmata, et al. 2000].

The two solutions, which use labeled and unlabeled sample documents and are proposed in previous research [Nigam, et al. 2000][Skarmata, et al. 2000], have their own limit for text categorization; they are useless without labeled sample documents. For example, Nigam's solution needs labeled documents to train Naïve Bayes initially. Skarmeta's solution also needs labeled ones to provide the classification system for the clustering of unlabeled documents with the ssAHC algorithm. These solutions characterize the utilization of both labeled and unlabeled documents for building classifiers; they also need the manual preparation of labeled documents before the main task of text categorization.

2.4.3 Most Common Approaches to Text Categorization

This section covers four existing approaches to text categorization: k-NN (k Nearest Neighbor), NB (Naïve Bayes), SVM (Support Vector Machine), and BP (Back Propagation). The first traditional approach, k-NN, is applicable to any strategy of encoding documents, once the similarity between two documents encoded in such a strategy is defined. Although NB, SVM, and BP are popular approaches to text

categorization [Mitchell 1997], these approaches require the representation of documents into numerical vectors which leads to the two main problems already discussed: huge dimensionality and sparse distribution. For each approach, this section will describe its general nature, its process for classifier training and document classification, and its advantages and disadvantages.

2.4.3.1 K Nearest Neighbor

The first approach to classifier training and document classification is k-nearest neighbor which is defined as the supervised learning algorithm with which novel data are classified, based on the majority of their neighborhoods of sample data. The first characteristic of this approach is that its process is simple, because only several training examples are requested in order to classify novel examples, instead of optimizing the parameters of classifiers in advance, using all the training examples. Its second characteristic is that it could be applied to any type of representation of documents, once the content similarity between documents is defined. It is applicable to two document encodings described in section 2.1. In this respect, this approach corresponds to single pass algorithm, in the context of text clustering. K Nearest Neighbor is a lazy supervised learning algorithm, because it learns sample data selectively after unseen data is given for their classification. Unlike other supervised learning algorithm, it does not learn all the sample data in advance. It belongs to the family of instance based learning algorithms, because it classifies data with reference to each of the sample data, rather than a summarized representation of the data [Mitchell 1997].

Figure 8 illustrates the process of maintaining the organization for each of the added documents using k nearest neighbor. Before this process begins, the documents contained in the organization should be encoded into bags of words, or numerical vectors. In figure 8, labeled sample documents refer to the documents contained in the organization and an unseen document refers to documents added after the initial organization was completed. For each of the given labeled sample documents, the similarity with the encoded unseen document is computed with one of the strategies described in section 2.1. Among them, k labeled documents with their highest similarities, are selected. The label of the unseen document is determined by voting. The document is then arranged into the cluster corresponding to its classified label in the given organization.

Input: an unseen document, encoded labeled sample documents, and the value of k
Step 1: Encode the unseen document into a bag of words, a numerical vector, or a string vector
Step 2: compute the similarity of each sample document with the unseen document
Step 3: get k most similar sample labeled documents
Step 4: label the unseen document by voting selected sample labeled documents
Step 5: arrange the document into the cluster corresponding to its classified label
Output: the classified label of the document

Figure 8. The Process of Classifier Training and Document Classification with K-NN

If k Nearest Neighbor is used to maintain the given organization of documents, both classifier training and document classification are performed on each added documents interactively: in figure 7, step 2 and step 3 correspond to classifier training and step 4 and step 5 correspond to document classification.

K Nearest Neighbor is easy to implement and understand because of its intuitional simplicity. In 1999, Yang showed in her direct and indirect evaluation[7] of 12 approaches to text categorization that three approaches, LLSF (Least Linear Square Fit), k-Nearest Neighbor, and Perceptron have best performance among them [Yang 1999]. Its performance on text categorization is comparable to SVM, BP, and NB on the standard test bed, Reuter 21578 [Sebastiani 2002]. Therefore, k-Nearest Neighbor is very popular not only for text categorization but also for other problems of pattern classification [Mitchell 1997].

A serious disadvantage of k-Nearest Neighbor is that it is not useful in a large organization of documents, since whenever an unseen document is given, the similarities of all sample documents with it should be computed. It takes a lot of time to perform the two phases involved in text categorization. As time goes, documents accumulates gradually in any textual information system, since more documents are added than deleted. With this approach, it is therefore, impossible to manage the organization of added documents in real time. Hence, k Nearest Neighbor is less practical in managing an organization of documents in a textual information system.

2.4.3.2 Naïve Bayes

The second approach to classifier training and document classification is NB (Naïve Bayes), the practical version of Bayesian learning method where the classification of data is achieved based on the maximum of the posterior probabilities of categories given the data. NB is a particular instance of Bayesian learning under the assumption that attributes are independent of each other.

An item of data, which is targeted for classification, is denoted by an ordered set of attributes, $a = \lfloor a_1 \quad a_2 \quad .... \quad a_{|a|} \rfloor$, and a set of given categories indicating a classification system is denoted by $C = \{c_1, c_2, ..., c_{|C|}\}$. The classification rule in Bayesian learning is expressed in equation (12),

$$\arg\max P(c_k \mid a) = \arg\max P(a \mid c_k) P(c_k) \quad (12)$$

where $P(c_k \mid a)$ is the posteriori probability of the category, c_k $1 \le k \le |C|$. Given an item of data, a, $P(a \mid c_k)$ is the probability density function of the item of data, a given the category c_k, and $P(c_k)$ is the priori probability of category, c_k. According to the classification rule expressed with equation (12), the category with the highest posteriori probability is assigned to the item of data, as the process of classifying data. In the completely balanced distribution over categories, the classification rule is simplified like in equation (13), because all priori probabilities of categories are identical.

$$\arg\max P(c_k \mid a) = \arg\max P(a \mid c_k) \quad (13)$$

In Naïve Bayes, the probability density function, $P(a \mid c_k)$, is expressed by the product of probability density functions of attributes like in equation (14), since it is assumed that all the attributes are independent of each other,

---

[7] In her study, direction evaluation means the process of evaluating such approaches by performing experiments directly, while indirect evaluation means the process of only brining results from other published literatures.

$$P(a \mid c_k) = \prod_{j=1}^{|a|} P(a_j \mid c_k) \quad (14)$$

where $P(a_j \mid c_k)$ is the probability density function of the attribute value, a_j $1 \le j \le |a|$, in category c_k. If equation (14) is assigned in equation (12), the classification rule in Naïve Bayes is expressed as in equation (15).

$$\arg\max P(c_k \mid a) = \arg\max P(c_i) \prod_{j=1}^{|a|} P(a_j \mid c_k) \quad (15)$$

Figure 9 shows the process of classifier training and document classification using Naïve Bayes. As illustrated in figure 9, the two phases are separated. A series of sample documents given as the input of classifier training in figure 9 indicates the documents contained in the organization. They are encoded using one of the three strategies of document encoding (bags of words, numerical vectors, or string vectors) in their preprocessing phase. For each class, the prior probability is estimated as the ratio of the number of sample documents belonging to the cluster to that of all the sample documents. For each unique attribute value contained in the given encoded sample documents, the probability density function given each cluster is estimated. The process of estimation is different depending on the strategy of encoding documents, so it will be explained later. The output of classifier training is a series of prior probabilities of classes and probability density functions of unique attribute values of document encodings given classes.

In the process of document classification, an unseen document and the output of classifier training are given as its input. The unseen document is encoded using the strategy of document encoding identical to the one used in classifier training. The posteriori probability of each cluster given the encoded document is computed by multiplying its priori probability and the product of the probability density functions of the attribute values contained in the encoded document given the cluster. The document is assigned to the cluster with the highest posterior probability. In document organization, we train a Naive Bayes classifier, where we define a class for each computed cluster. The output of document classification is the identifier of that cluster.

Classifier Training
Input: A Series of Sample Documents
Step 1: Encode these sample documents into bags of words, numerical vectors, or string vectors
Step 2: For each cluster
Step 2-1: estimate the priori probability of the cluster
Step 2-2: For each attribute value
Step 2-2-1: estimate the probability density function of the attribute value
Output: the list of priori probabilities and the list of probability density functions

Document Classification
Input: An unseen document, the list of priori probabilities, and the list of probability density functions
Step 1: Encode the unseen document into the same structured data to that in classifier training
Step 2: For each cluster
Step 2-1: posteriori probability of the cluster = the priori probability of the cluster
Step 2-2: For each attribute value in the unseen document
Step 2-2-1: get the corresponding probability density function from the list
Step 2-2-2: posteriori probability of the cluster = posteriori probability of the cluster * probability density function
Output: the cluster corresponding to the maximum of posteriori probabilities

Figure 9. The Process of Classifier Training and Document Classification with Naive Bayes

The probability density function of each unique attribute value is estimated differently depending on the strategy used for encoding documents. In the first strategy of document encoding, an encoded document is a bag of words, and is denoted by $d_i^b = \{w_{i1}, w_{i2}, ..., w_{il}\}$ as an unordered set of words. The number of documents in the cluster, c_k, is denoted by $|D_k|$, and the number of documents including the word, w_{ij} in the cluster, c_k, is denoted by $|D_k(w_{ij})|$. The probability density function of the word, w_{ij}, given the cluster, c_k, $P(w_{ij} \mid c_k)$ is estimated using equation (16). Equation (16) indicates the ratio of the number of documents including the word, w_{ij}, to the total number of documents within the cluster, c_k.

$$P(w_{ij} \mid c_k) = \frac{|D_k(w_{ij})|}{|D_k|} \quad (16)$$

The classification rule of the encoded document, $d_i^b = \{w_{i1}, w_{i2}, ..., w_{il}\}$, in the first strategy is defined by equation (17)

$$\arg\max P(c_k \mid d_i^b) = \arg\max P(c_k) \prod_{j=1}^{l} P(w_{ij} \mid c_k) = \arg\max P(c_k) \prod_{j=1}^{l} \frac{|D_k(w_{ij})|}{|D_k|} \quad (17)$$

In the second strategy of document encoding, an encoded document is a numerical vector, and is denoted by $d_i^f = [\phi_i(w_1) \quad \phi_i(w_2) \quad .... \quad \phi_i(w_n)]$. Its elements were already explained in section 2.1. Each attribute value is denoted by $a_j = \phi_i(w_j) \quad 1 \le j \le n$. The number of encoded documents which have the value of the j th attribute, $a_j = \phi_i(w_j)$ within the cluster, c_k, is denoted by $|D_k(a_j = \phi_i(w_j))|$. The probability density

function of such attribute value, $a_j = \phi_i(w_j)$ given the cluster c_k $P(a_j = \phi_i(w_j) \mid c_k)$ is estimated using equation (18).

$$P(a_j = \phi_i(w_j) \mid c_k) = \frac{|D_k(a_j = \phi_i(w_j))|}{|D_k|} \quad (18)$$

Therefore, the classification rule of the document encoded in the numerical vector, $d_i^f = [\phi_i(w_1) \quad \phi_i(w_2) \quad .... \quad \phi_i(w_n)]$, is defined by equation (19).

$$\text{argmax}P(c_k \mid d_i^f) = \text{argmax}P(c_k)\prod_{j=1}^{n} P(a_j = \phi_i(w_j) \mid c_k) = \text{argmax}P(c_k)\prod_{j=1}^{n} \frac{|D_k(a_j = \phi_i(w_j)|}{|D_k|} \quad (19)$$

Although Naïve Bayes is the most popular approach to text categorization, it has its own disadvantage. If one probability of a particular attribute, given the cluster, c_k, $P(a_j \mid c_k)$, is zero, the posterior probability of the cluster, c_k, given the item of data, $a = [a_1 \quad a_2 \quad .... \quad a_{|a|}]$ is zero. If a strange document including words beyond the domain of sample documents is added in document classification using NB, it can not be classified into one of clusters, since several of its posterior probabilities of clusters may be zeros.

In order to address the disadvantage of its original version, the m-estimate is used to estimate probabilities involved in Naïve Bayes [Mitchell 1997]. The m-estimate prevents the probabilities from being close to zero by restricting the minimum of estimates of the probabilities with a particular constant, instead of zero. If the m-estimate is applied to Naïve Bayes given as a text classifier, equation (19) computing an estimation of a probability of the value w_{ij} of an attribute, a_j given a category, c_k is modified into equation (20),

$$P(a_j = w_{ij} \mid c_k) = \frac{|D_k(a_j = w_{ij})| + mp}{|D_k| + m} \quad (20)$$

where m and p are given as its parameters arbitrary; m is given as an arbitrary integer, and p is given as a normalized continuous value between zero and one. In the parameters, m indicates the number of virtual additional training examples, and p indicates the probability that a virtual training example has such attribute value. From equation (20), the minimum value among estimations of the probabilities is given as the value, $\frac{mp}{|D_k| + m}$, instead of zero. Hence, the classification rule of the Naïve Bayes using m-estimate is given as equation (21).

$$\text{argmax}P(c_k \mid d_i^f) = \text{argmax}P(c_k)\prod_{j=1}^{n} P(a_j = w_{ij}) \mid c_k) = \text{argmax}P(c_k)\prod_{j=1}^{n} \frac{|D_k(a_j = w_{ij})| + mp}{|D_k| + m} \quad (21)$$

2.4.3.3 Support Vector Machine

The third approach to classifier training and document classification is SVM (Support Vector Machine). This approach is a kernel based supervised learning algorithm for a

binary classification problem [Muller, et al. 2001]. SVM defines two hyper planes corresponding to the positive and negative classes in the linearly separable feature space mapped from the given original space using a kernel function with maximal margin between them. Support vectors correspond to the training examples that influence the determination of the two hyper planes as the boundaries of the two classes. This approach has its advantages both theoretically and practically [Hearst 1998]. The performance of SVM in text categorization and other classification problems in image processing and bioinformatics are better than other approaches [Sebastiani 2002][Joachims 1998][Cristianini and Shawe-Taylor 2000].

An item of data, which is targeted for classification, is represented into a numerical vector denoted by $\mathbf{x}$. This vector is classified using equation (22),

$$f(\mathbf{x}) = sign((\mathbf{x} \cdot weight) + b) \text{ (22)}$$

where *weight* is the weight vector consisting of the parameters of classification, b is the threshold, and $sign(\cdot)$ is the function determining that if the input is more than zero, the output is positive, and otherwise, the output is negative. Equation (22) expresses the classification rule of a hyper plane for binary classification. Its output is positive or negative. The weight vector, *weight*, is computed using the linear combination of training examples expressed as in equation (23),

$$weight = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i \text{ (23)}$$

where α_i $1 \le i \le N$ is the Lagrange multiplier corresponding to the training example, $\mathbf{x}_i$, and N is the number of training examples. Therefore, equation (22) is converted into equation (24), the equation of the hyper plane based on Lagrange multipliers, using equation (23).

$$f(\mathbf{x}) = sign(\sum_{i=1}^{N} \alpha_i (\mathbf{x} \cdot \mathbf{x}_i) + b) \text{ (24)}$$

Therefore, in equation (24), the parameters for classification in SVM are Lagrange multipliers, instead of the elements of the weight vector.

The basic idea underlying in SVM is to build the linear classifier expressed by equation (24) with the maximal margin, in the linearly separable space mapped from the original space, as illustrated in figure 10 [Muller, et al. 2001]. The vector in the mapped space corresponding to the original vector, $\mathbf{x}$, is denoted by $\psi(\mathbf{x})$. The substitution of two vectors, $\psi(\mathbf{x})$ and $\psi(\mathbf{x}_i)$, for the two vectors, $\mathbf{x}$ and $\mathbf{x}_i$, in equation (24) leads to equation (25).

$$f(\mathbf{x}) = sign(\sum_{i=1}^{N} \alpha_i (\psi(\mathbf{x}) \cdot \psi(\mathbf{x}_i)) + b) \text{ (25)}$$

The inner product of $\psi(\mathbf{x})$ and $\psi(\mathbf{x}_i)$ is defined as a kernel function of the two original vectors, $\mathbf{x}$ and $\mathbf{x}_i$, $\psi(\mathbf{x}) \cdot \psi(\mathbf{x}_i) \equiv K(\mathbf{x} \cdot \mathbf{x}_i)$. The goal of the kernel function is to compute the inner product of $\psi(\mathbf{x})$ and $\psi(\mathbf{x}_i)$ without mapping $\mathbf{x}$ and $\mathbf{x}_i$ into $\psi(\mathbf{x})$ and $\psi(\mathbf{x}_i)$ individually. Using a kernel function, equation (25) is converted to equation (26).

$$f(\mathbf{x}) = sign(\sum_{i=1}^{N} \alpha_i (K(\mathbf{x} \cdot \mathbf{x}_i)) + b) \text{ (26)}$$

Therefore, equation (26) is the general form of a SVM classifier for a binary classification problem. As follows, there are several types of kernel functions.

- identity kernel function: $K(\mathbf{x}\cdot\mathbf{x}_i)\equiv\mathbf{x}\cdot\mathbf{x}_i$
- linear kernel function: $K(\mathbf{x}\cdot\mathbf{x}_i)\equiv a(\mathbf{x}\cdot\mathbf{x}_i)+b$
- polynomial kernel function: $K(\mathbf{x}\cdot\mathbf{x}_i)\equiv((\mathbf{x}\cdot\mathbf{x}_i)+\theta)^{\kappa}$
- exponential kernel function: $K(\mathbf{x}\cdot\mathbf{x}_i)\equiv\exp(K_1(\mathbf{x}\cdot\mathbf{x}_i))$
- Gaussian kernel function: $K(\mathbf{x}\cdot\mathbf{x}_i)\equiv\exp\left(\frac{-\|(\mathbf{x}-\mu)(\mathbf{x}_i-\mu)\|}{\sigma^2}\right)$

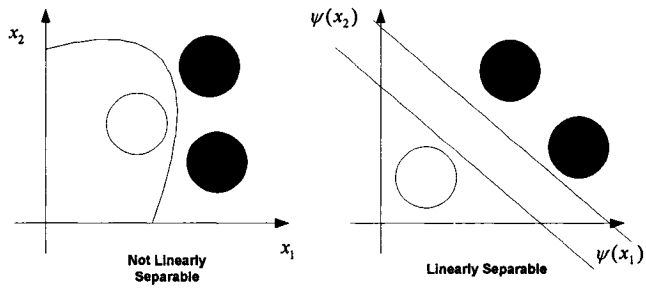

Figure 10. Mapping Vector Space in SVM

The learning in SVM is the process of optimizing the Lagrange multipliers, α_i as given as the parameters of classification. This process is performed in the phase, classifier training, when SVM is used in the four-phases-scenario. If the distribution of training data is linearly separable in the mapped space, the object function and its constraint are defined by equation (27), in order to optimize the Lagrange multipliers, α_i. Keeping the given constraint that all the Lagrange multipliers should be non negative, their optimization corresponds to minimizing the object function, $W(\alpha)$.

$$\begin{cases} W(\alpha)=\sum_{i=1}^{N}\alpha_i-0.5\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j K(\mathbf{x}_i\cdot\mathbf{x}_j)y_i y_j \\ \text{The constraint}:\alpha_i\geq 0 \end{cases} \quad (27)$$

In equation (27), y_i and y_j are respective target categories of two vectors of training data, $\mathbf{x}_i$ and $\mathbf{x}_j$. If the distribution of training examples in the mapped space is not linearly separable, the object function for its minimization and its constraint are modified into equation (28), in order to allow a training error for each example.

$$\begin{cases} W(\alpha)=\sum_{i=1}^{N}\alpha_i-0.5\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j\left(K(\mathbf{x}_i\cdot\mathbf{x}_j)+\frac{\delta_{ij}}{C}\right)y_i y_j \quad \delta_{ij}=\begin{cases}1 \text{ if } i=j\\ 0 \text{ otherwise}\end{cases} \\ \text{The constraint: } 0\leq\alpha_i\leq C \end{cases} \quad (28)$$

In equation (28), C is the capacity of the Lagrange multipliers and is given as an input parameter.

Some typical strategies for optimizing Lagrange multipliers during the process of training SVM, are chunking [Vapnik 1982], Osuna's approach [Osuna, et al. 1997], and SMO (Sequential Minimal Optimization) [Platt 1998]. Among them, SMO is adopted as the approach to classifier training in this study, because it performs the optimization fastest and most efficiently with respect to system resource [Platt 1998]. In SMO, the

Lagrange multipliers and the threshold are initialized to zero. Two training examples of which at least one is misclassified using equation (26) with current Lagrange multipliers are selected and their corresponding Lagrange multipliers are modified using the analytical optimization. This process is repeated until there is no misclassified training example any more. The detail of the algorithm is described in the literature, [Platt 1998]. One of the main issues about SVM is how to optimize these Lagrange multipliers to minimize the object function and its constraint presented in equation (28).

Figure 11 illustrates the process of classifier training and document classification using SVMs. In the process of classifier training, a series of sample documents, which are included in the organization, is given as its input, and the capacity of Lagrange multipliers, C, and a kernel function are given as the parameters of SVMs. As a preprocessing step, all the sample documents are encoded into numerical vectors. For each cluster, a list of these encoded sample documents is partitioned into positive examples, indicating the members of the cluster, and negative examples, indicating the nonmembers of the cluster. The SVM corresponding to each cluster learns from these positive examples and negative examples; its Lagrange multipliers and its threshold are optimized in classifier training. The final output of this process is a series of SVMs corresponding to the given clusters with their associated optimized Lagrange multipliers and their thresholds.

Classifier Training
Input: A Series of Sample Documents, the Capacity of Lagrange Classifiers, and a kernel function
Step 1: Encode these sample documents into numerical vectors
Step 2: For each cluster
Step 2-1: Build the SVM corresponding to the cluster
Step 2-2: Positive Examples ← Documents belonging to the cluster
Step 2-3: Negative Examples ← Documents not belonging to the cluster
Step 2-4: Train the SVM using SMO with positive examples and negative examples
Output: SVMs corresponding to clusters with optimized Lagrange multipliers and thresholds of each SVM

Document Classification
Input: An unseen document, the list of SVMs corresponding to clusters
Step 1: Encode the unseen document into a numerical data
Step 2: For each SVM
Step 2-1: classify the encoded document in positive or negative
Step 3: arrange the document into the cluster where it is classified into its positive class
Output: the cluster identifier where it is arranged

Figure 11. The Process of classifier training and document classification using SVMs

In the process of document classification, an unseen document and the output of classifier training are given as its input. It is also encoded into a numerical vector and transferred to each SVM as its input vector. The document is arranged into the cluster classified by SVM as positive. If more than two SVMs classify it as a positive example, it is arranged into the cluster with its highest value of $\sum_{i=1}^{N} \alpha_i (K(\mathbf{x} \cdot \mathbf{x}_i)) + b$, among them, since these value indicates the degree of membership in each cluster.

Although SVM has its theoretical and practical advantages, as mentioned above, it has the disadvantage of being applicable to binary classification problems, only. When SVM

is used in a multiple classification problem, it should be applied to such problems not as a single SVM but as a committee consisting of several SVMs, because a multiple class classification problem should be partitioned into several binary classification problems. Therefore, SVM is not suitable for a big organization including many clusters. SVM is thus not recommendable for a DDO system, because documents belonging to new topics may be added continuously and the number of clusters increase.

2.4.3.4 Back Propagation

The fourth approach to classifier training and document classification is BP (Back Propagation), called MLP (Multiple Layers Perceptron). MLP was proposed initially by Rumelhart in 1986 [Rumelhart, et al. 1986]. The reason for referring to MLP as BP is that its parameters for classification are optimized from output layer to input layer, in the backward direction, and this approach will be mentioned as Back Propagation in this proposal. Among supervised neural networks, Back Propagation is the most popular model, because of its easy implementation, its tolerance to noise in the training data, and its application to a wide scope of problems.

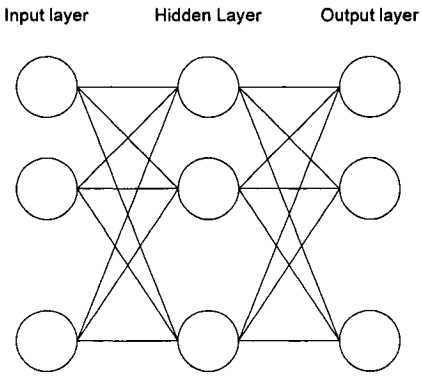

Figure 12. The Architecture of Back Propagation

Figure 12 illustrates the architecture of back propagation. There exist three layers, input layer, hidden layer, and output layer in the architecture. Two adjacent layers are completely connected to each other, and each connection indicates a weight as a parameter for classification. Learning in BP is the process of optimizing each weight corresponding to each connection to minimize the error between the target category and the classified category of each training example. In the design of BP for a particular classification problem, the number of input nodes corresponds to the dimension of each numerical vector representing a training example; the number of output nodes corresponds to the number of categories; and the number of hidden nodes is determined arbitrary. Note that there is no standard rule of determining the number of hidden nodes. In the role of each layer for classification problems, the input layer receives each input vector for classification or learning, the hidden layer defines the quadratic boundary between categories, and the output layer generates the result of classification.

Table 1 defines the notation and the output value of each node in each layer of the back propagation network. Each column of table 1 indicates each layer; the first row presents the notation of the nodes in each layer as a set, and the second row presents the equations for computing the value of each node, when an input vector denoted by

$\mathbf{x} = [x_1 \quad x_2 \quad \ldots \quad x_n]$ is given. Since the number of input nodes should be identical to the dimension of the input vector, the value of $|\mathbf{i}|$ is n. The value of an input node is its corresponding element of an input vector. The value of a hidden node or an output node is the output of the sigmoid function of the linear combination of the values of the nodes in the previous layer multiplied by the weights which connects from all the the nodes in the previous layer to itself. The output of the sigmoid function is a normalized value from zero to one. In the weight, $weight_{kj}$, the subscript, k, indicates the index of the destination node and the subscript, j indicates that of the source node. The process of classifying the input vector, $\mathbf{x}$, is achieved in the forward direction, from the input layer to the output layer. The number of output nodes is the same as the number of the given categories, because each output node corresponds to each category. The output node, where a value is closest to one, indicates that its corresponding category is the classified category of the input vector, $\mathbf{x}$.

Table 1. The Definition of Nodes in each Layer in BP

	Input Layer	Hidden Layer	Output Layer
The Notation of Nodes	$\mathbf{i} = \{i_1, i_2, \ldots, i_{\|\mathbf{i}\|}\}$	$\mathbf{h} = \{h_1, h_2, \ldots, h_{\|\mathbf{h}\|}\}$	$\mathbf{o} = \{o_1, o_2, \ldots, o_{\|\mathbf{o}\|}\}$
The Value of Nodes	$i_k = x_k$	$h_k = \frac{1}{1 - \exp(-\sum_{j=1}^{\|\mathbf{i}\|}(i_j weight_{kj})}$	$o_k = \frac{1}{1 - \exp(-\sum_{j=1}^{\|\mathbf{h}\|}(h_j weight_{kj})}$

In BP, two weight matrices are defined; one is from the input layer to the hidden layer, and the other is from the hidden layer to the output layer. The supervised learning algorithm, BP, optimizes the weights included in these two weight matrices, as its learning process, for the minimization made on the error of training data. The error function is defined as the object function that is to be minimized when modifying these weights. The objective function representing the training error is expressed by equation (29),

$$E = \frac{1}{2}\sum_{j=1}^{|\mathbf{o}|}(t_j - o_j)^2 \quad (29)$$

where t_j is the target value of the input vector, $\mathbf{x}$, corresponding to a category, and is given as a binary value, zero or one. One indicates that the input vector, $\mathbf{x}$, is a member of the corresponding category, and zero indicates that it is not a member. Note that o_j is given as a continuous value between zero and one, unlike the target value, t_j.

Table 2 presents two weight matrices, WEIGHT_{oh} indicating the connections between the hidden layer and the output layer, and WEIGHT_{hi} indicating those between the input layer and the hidden layer together with the rules for updating these weights to minimize the object function of equation (29). Two equations in the bottom row of table 2 are derived by differentiating equation (29) with the weight, using the chain rule. The optimization of all the weights of BP is achieved using the gradient descent method based on the slope of the error surface. The detailed process of deriving these rules for updating weights is described in any text book about neural networks (e.g. [Haykin 1994]). At first, these weights are initialized into values around zero, randomly. They are updated in the

backward direction, from the output layer to the input layer. In both equations in table 2, η indicates the learning rate given as a parameter for this approach. Too large a value of it leads to too much fluctuation in training BP, and too small a value leads to slow learning. Its value is usually set between 0.1 and 0.3.

Table 2. The Definition of Weights of BP

	Hidden Layer → Output Layer	Input Layer → Hidden Layer
Weight Matrix	$\text{WEIGHT}_{oh} = \begin{bmatrix} weight_{11} & \dots & weight_{1\|h\|} \\ \dots & \dots & \dots \\ weight_{\|o\|1} & \dots & weight_{\|o\|\|h\|} \end{bmatrix}$	$\text{WEIGHT}_{hi} = \begin{bmatrix} weight_{11} & \dots & weight_{1\|i\|} \\ \dots & \dots & \dots \\ weight_{\|h\|1} & \dots & weight_{\|h\|\|i\|} \end{bmatrix}$
Update Rule	$\Delta weight_{kj} = \eta(t_k - o_k)(1-o_k)o_k h_j$	$\Delta weight_{kj} = \eta \sum_{r=1}^{\|o\|}(t_r - o_r)(1-o_r)o_r w_{rk}(1-h_k)i_j$

Figure 13 illustrates the process of classifier training and document classification using BP. In the process of classifier training, a series of sample documents is given as its input, and the learning rate and iteration number are given as the parameters of BP. During the preprocessing phase, these sample documents are encoded into numerical vectors. The architecture of BP is designed, so that the number of input nodes is the dimension of these numerical vectors encoding sample documents, the number of hidden nodes is set arbitrarily, and the number of output nodes is the number of clusters in the document organization. All the weights of BP are initialized into random values around zero. For each numerical vector encoding a sample document, the values of the output nodes are computed using the equations included in table 1. The weights are updated in the backward direction using the equations presented in table 2. The process of computing the values of the output nodes and updating all the weights of BP for each sample document is repeated with the iteration number given as a parameter. The final output of classifier training using BP is the two matrices of optimized weights, WEIGHT_{oh} and WEIGHT_{hi}.

In the process of document classification, an unseen document and two matrices of optimized weights, WEIGHT_{oh} and WEIGHT_{hi}, as the output of classifier training, are given as its input. The values of the output nodes to the numerical vector encoding the unseen document are computed using the equations presented in table 1. The document is arranged into the cluster corresponding to the output node, whose value is highest (closest to one).

We need to consider not only its advantages but also its disadvantages. Its first disadvantage is that it requires far more time for learning than SVM and NB. But Jacobs proposed a variation which speeds up learning, using an adaptive learning rate with a momentum term, as a solution to this disadvantage [Jacobs 1988]. Its second disadvantage is that the process of optimizing the weights of BP may fall into a local minimum of the error surface, because their optimization is achieved using gradient descent based on a slope of the error surface. But Yao applied a genetic algorithm in optimizing the weights instead of gradient descent to avoid falling into a local minimum [Yao 1993]. In spite of these disadvantages, BP is the most popular supervised neural networks, and has been used in various domains of pattern classification problems.

Classifier Training
Input: A Series of Sample Documents, Learning Rate, and Iteration Number
Step 1: Encode these sample documents into numerical vectors
Step 2: Design the architecture of BP
Step 3: Initialize weights into random values around zero
Step 4: Repeat step 3-1 with the number of iteration
Step 4-1: For each encoded sample document
Step 4-1-1: compute the values of output nodes of the encoded document with the current weight
Step 4-1-2: update weights based on the error
Output: Two optimized weight matrices

Document Classification
Input: An unseen document and two optimized weight matrices
Step 1: Encode the unseen document into a numerical vector
Step 2: Compute the values of the output nodes of the encoded document with the current weight
Step 3: arrange the document into the cluster whose output node has the highest value (closest to one)
Output: the cluster identifier where it is arranged

Figure 13. The Process of Classifier Training and Document Classification using BP

2.5 Class Imbalance Problem and Resampling

This section mentions the class imbalance problem and explores previous research on resampling methods as solutions to the problem. Whatever training data we use for training supervised learning algorithms, we can not avoid class imbalance problem in almost all application areas [Chawla et al 2004]. In this section, we will introduce the class imbalance problem in supervised learning tasks and resampling as a solution to the problem, and survey previous research on resampling methods.

Class imbalance problem refers to the performance loss of classification from imbalanced distributions of training examples over classes. If any classifier learns from imbalanced training examples, it builds a strong bias toward the majority class. The classifier tends to classify unseen examples into the majority class with the highest probability; many unseen examples may be misclassified into the majority class. Since the class imbalance problem degrades the classification performance so, we need to solve the problem before training. This section adopts resampling as the solution to the problem, and surveys the previous research on it.

Resampling refers to the process of making an imbalanced distribution of training examples into an almost balanced distribution by generating additional examples or removing some existing ones. There are two simple resampling methods, simple oversampling and simple undersampling. Simple oversampling is the process of adding additional training examples to the minority classes randomly by copying some of the existing ones. It is more stable than simple undersampling although it is less effective for the minority classes; oversampling does not degrade the performance of classification toward the majority classes, although it improves the performance of classification toward the minority classes less effectively than simple undersampling. Simple undersampling is the process of removing some of the existing training examples at random in the majority classes. It is more risky than simple oversampling, since it degrades the performance of classification toward the majority classes, although it is more effective in the minority classes than simple oversampling.

Estabrooks and his colleagues proposed a combination of resampling methods combining ten degrees of oversamplings and ten degrees of undersamplings in 2004 [Estabrooks, et al. 2004]. The method builds training sets corresponding to these twenty resampling schemes and builds the corresponding number of classifiers with these training sets. These classifiers choose the class of unseen examples using a voting scheme. They applied this resampling method to several classification problems, including text categorization on Reuter 21578, and showed that its performance is better than simple oversampling or simple undersampling, and as well as Adaboost, a ensemble based model of machine learning algorithms.

The consideration of small disjuncts is necessary to propose a state of the art resampling method, because small disjuncts are an important factor of the classification error [Weiss 2003][Weiss 1998]. A disjunct is the subconcept of a class that was built by learning known training examples. A disjunct's size is measured by the number of covered training examples. A small disjunct is the subconcept covering a small number of training examples. G. Weiss researched small disjuncts systematically, and proved they are very harmful to classification performance [Weiss 2003][Weiss 1998]. In 2004, Jo and Japkowicz showed empirically that the performance loss of classifiers trained with training examples with high class imbalance is not directly related to the class imbalance itself, but derives from the small disjuncts that were built from training classifiers with small classes [Jo and Japkowicz 2004]. These previous studies stated that small disjuncts are very important factors of classification error: they suggest the development of new resampling methods.

In 2000, Chawla and his colleagues proposed SMOTE (Synthetic Minority Oversampling Technology) as a resampling method [Muller, et al. 2001]. Note that in their proposed approach, examples oversampled for a minority class are not duplicates of existing examples. An example in the minority is taken, and its several nearest neighbors in that class, are selected in SMOTE. Synthetic examples are generated by taking a difference between the example and one of its nearest neighbors selected at random. This process is repeated until the achievement of a balanced distribution of training examples. The study showed that SMOTE improved classification performance more than oversampling by copying existing examples in the minority class [Muller, et al. 2001].

Chapter 3 DDO System and its Components

The previous chapter described the existing representations of documents and the existing approaches to text categorization and text clustering, and surveyed previous research on topics involved in implementing DDO systems. This chapter describes the concepts of DDO systems in the first section, the roles of the components of DDO systems in the second section, and the cycle of the creation mode and the maintenance mode as a coordination of the involved components in the third section.

3.1 Concept and Implementation of DDO

This section explains document organization in general and then describes dynamic document organization, and its implementation, in the conceptual view. Before being able to understand the concept of DDO, we need to understand the concept and the significance of generic document organization in managing documents. This will be described in detail in the first subsection. In the second subsection, the two opposite concepts of SDO and DDO are discussed, and note that the previous research on text clustering and text categorization has assumed SDO, rather than DDO. The last subsection will present the two scenarios for the three main tasks of DDO: the three-phases-scenario and the four-phases-scenario.

3.1.1 Document Organization

This subsection is concerned with the concept of generic document organization, and presents some examples. As defined in section 1.1, document organization is the process of arranging documents based on their contents. Although document organization was already mentioned in previous research on text clustering [Kohonen, et al. 2000][Kaski, et al. 2000], this study defines the concept more formally, using the following four conditions.

Definition 1. Document Organization

- A document organization is a collection of documents that are partitioned into several sub-collections known as clusters.
- Each cluster must include documents similar in contents.
- Every document in the collection must belong to one or several of clusters[8].
- Each cluster must be labeled with a name that is unique and reflects the contents of the included documents.

Therefore, the output of a document organization is the list of labeled clusters that abide by the above conditions.

Figure 14 shows an example of a document organization of a collection of news articles. The organization illustrated in figure 14 results in three clusters, 'business', 'health', and 'computers'. In this thesis, the list of these cluster names is called a classification system.

[8] If every document belongs to only one of clusters, it is an exclusive version of DDO. Otherwise, it is an overlapping version of DDO.

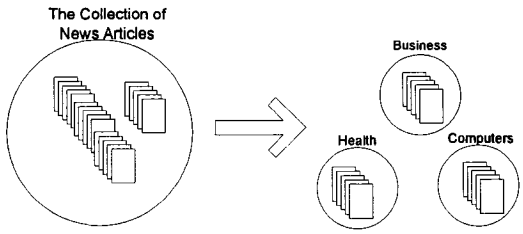

Figure 14. An Example of Document Organization in News Articles

Figure 15 presents another example of a document organization of a collection of email messages. The figure shows that email messages are organized into three clusters, 'advertisement', 'meeting', and 'call for paper'.

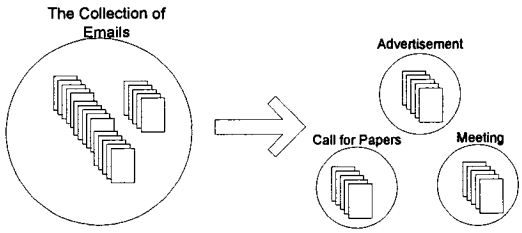

Figure 15. An Example of Document Organization in Email Messages

A document, in our system, is an electronic article written entirely in a natural language; examples of documents are news articles, email messages, research papers, project proposals, patent documents, and so on. Although electronic articles written in more than two languages are also documents, they are excluded in this research; only electronic articles written in English are targeted in this research. The organization of documents enables users to access them easily. If documents are piled regardless of their contents, users must access all of them sequentially and individually, to find the one they are interested in. In every textual information system, documents are always managed through the maintenance of their organization, whether it is manual or automatic.

In the early ages of textual information systems, documents were organized manually; administrators read and arranged them individually into their corresponding clusters. This manual organization of documents cost very much time when dealing with a massive collection of documents and required experts in each domain. In order to automate document organization completely or partially, text categorization and text clustering were proposed afterwards. In text categorization, manual tasks, such as the predefinition of a classification system and allocation of sample documents to each cluster, are still necessary for document organization. In text clustering, reclustering following every manipulation of a collection of documents is impractical since its computation is very expensive, despite the fact that text clustering does not require any manual task. Therefore, this study proposes a circular integration of text clustering, cluster identification, and text categorization, to automate the document organization completely even in a variable collection of documents.

3.1.2 Static vs Dynamic Document Organization

This section will explain the concept of DDO in detail by contrasting its properties to those of SDO. Table 3 summarizes the properties of the two types of document organization, SDO and DDO. In table 3, the left column and the right column indicate SDO and DDO, respectively. Each row of table 3 indicates an aspect of their properties.

Table 3. The Properties of SDO and DDO

	SDO	DDO
Collection of Documents	Constant	Variable
Users' Manipulation	Excluded	Considered
Classification System	Permanent	Temporary
Implementation	Simple	Complicated
Real Situation	Impractical	Practical

SDO (Static Document Organization) is defined as a document organization which is always constant for a collection of documents. In this type of document organization, it is assumed that a collection of documents is constant permanently; users' manipulations on the collection are excluded. Once this classification system is defined through document organization, it exists permanently. Since SDO consists of building the document organization once and maintaining it continually, its implementation is simpler than DDO's. Because, in reality, a collection of documents always changes due to users' manipulation and because SDO ignores this fact, SDO is not practical in our real world.

DDO (Dynamic Document Organization) is defined as the document organization which is automatically adaptable to real situations taking place in a collection of documents. In this type of document organization, it is assumed that the collection of documents always varies; users' manipulations on it are considered. Since documents are reorganized depending on the situations that occur in the collection of documents, the classification system exists only temporarily. Because DDO consists of building the document organization and maintaining it, the conditions for its reorganization should be defined, and integrated in the system. Therefore, its implementation is more complicated than SDO's. Such conditions will be described in detail in section 3.3. Since the organization of documents is changed automatically according to real situations in the collection of documents, it is practical in our real world.

DDO should satisfy all the following conditions

- *If a sufficiently large number of documents belonging to new topics are added, corresponding clusters should be created.*

After building an organization of documents, users may add documents belonging to topics out of the organization. In this case, new cluster corresponding to topics should be added to the given organization of documents.

- *If documents are added to one of the existing clusters continually, this cluster should be partitioned into several clusters.*

After building an organization of documents, users may add documents belonging to some existing clusters. Since such clusters become very large, they should be partitioned in order to increase their cohesions.

- *If documents are continually deleted from one of existing clusters, the cluster should be merged with another cluster.*

After building an organization of documents, users may delete documents which are not necessary any more. If such clusters become sparse from such a continuous deletion, they need to be merged with other clusters.

Previous research proposed the integration of text categorization and text clustering under the concept of SDO [Wu, et al. 2001][Skarmata, et al. 2000]. As mentioned above, SDO is not practical in our real world. This study proposes strategies for implementing DDO, in order to develop more practical textual information systems.

3.1.3 Two Scenarios for DDO Implementation

This study proposes two scenarios for implementing DDO, the three-phases-scenario and the four-phases-scenario, and this section will describe them generally. In this subsection, the three-phase-scenario is presented in figure 16 and the four-phase-scenario is presented in figure 17. In table 4, this subsection summarizes the difference between the two scenarios. In table 5, all versions of our implementation of DDO are presented depending on the machine learning algorithm and the type of document encoding.

The three-phases-scenario illustrated in figure 16, consists of text clustering, cluster identification, and document classification. The assumption underlying both scenarios is that an initial collection of documents is empty. If a sufficient number of documents are piled into the collection, the system shifts to 'creation' mode - in a first phase, text clustering is performed to build an initial document organization. The number of documents initiating text clustering is set as an option. The first phase, "text clustering", results in a list of unlabeled clusters which includes documents similar in contents, together with the prototypes of such clusters. Each cluster is then labeled through cluster identification, which is the next phase of our system set to 'creation' mode. The process of cluster identification will be described in detail in section 3.2.

The creation mode of DDO systems generates clusters and their prototypes through the first phase, "text clustering" and their identifiers through the second phase, "cluster identification". The role of the maintenance mode of DDO systems is to arrange documents, which users add later, into their appropriate clusters. In the three-phases-scenario, tasks involved in the maintenance mode are performed in the phase, "document classification", in figure 16. Note that clustering algorithms generate not only clusters of similar objects but also prototypes which represent clusters. For example, the single pass algorithm generates prototypes of clusters by assigning its first object to each cluster when it is created initially, while Kohonen Networks and k means algorithm generate prototypes by updating weight vectors called prototype vectors. In DDO systems of the three-phases-scenario, similarities of an added document with prototypes of clusters are computed and the document is arranged into the cluster corresponding to its prototype with the highest similarity.

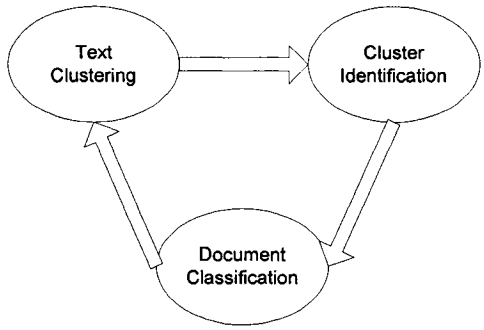

Figure 16. The three-phases-frame of implementing DDO

Figure 17 illustrates the four-phases-scenario which is identical to the three-phases-scenario except for the addition of the phase, "classifier training", between "cluster identification" and "document classification". The idea underlying this scenario is to improve the robustness of classifying added documents by taking into consideration all the existing documents, rather than only a few representative ones considered as cluster prototypes. The role of the "classifier training" phase is to build classification rules or to define classification equations, using supervised learning algorithms. Note that the previous two phases, "text clustering" and "cluster identification", generate labeled documents for the phase, "classifier training". Therefore, in this scenario, two phases, "classifier training" and "document classification" are involved in maintaining the organization of documents.

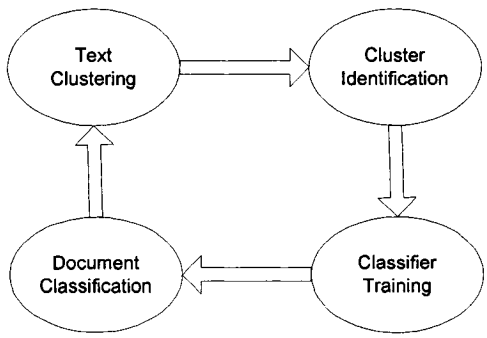

Figure 17. The four-phases-frame of implementing DDO

Table 4 summarizes the differences between the two scenarios for implementing DDO that were presented in figure 16 and figure 17. In table 4, the middle column indicates the properties of the three-phases-scenario and the right column indicates those of the four-phases-scenario. The phase, "text clustering", in both scenarios corresponds to unsupervised learning. Since the labeled documents that the two phases, "text clustering" and "cluster identification", generate are used to train classifiers, the phase, "classifier training", in the four-phases-scenario corresponds to supervised learning. In the three-phases-scenario, the classification of added documents is performed instantly as soon as the organization of documents is built, while, in the four-phases-scenario, it is delayed until the phase, "classifier training", is finished.

Table 4. The difference between three-phases-frame and four-phases-frame

Properties	Three-phases-frame	Four-phases-frame
Involved Tasks	Text Clustering Cluster Identification Document classification	Text Clustering Cluster Identification Classifier Training Document classification
Learning Algorithms	Unsupervised Learning	Unsupervised and Supervised Learning
References to Classification	Prototypes of Clusters	Classification Rules Classification Parameters
Classification Robustness	Less	More
Classification Delay	No	Yes

Table 5 specifies the supervised and unsupervised learning algorithms applied to the tasks involved in implementing DDO in both scenarios. In table 5, the left column indicates document encodings, where machine learning algorithms in the corresponding row are applicable. For example, the second row in the table indicates that documents should be represented into numerical vectors in order to apply Kohonen Networks to text clustering and SVM, and Back Propagation to text categorization. The third row indicates that the single pass algorithm, Naïve Bayes, and k-nearest neighbor are applicable to text clustering and text categorization in any type of document encodings. Each of the machine learning algorithms in the third row has three versions depending on the type of document encoding. Approaches in the last row are the new neural networks, NTSO (Neural Text Self Organizer) [Jo and Japkowicz 2005] and NTC (Neural Text Categorizer) [Jo 2004], that use an alternative representation of documents to numerical vectors. In this study, as illustrated in table 5, the three-phases-scenario and the four-phases-scenario have five versions and nine versions of DDO implementation scenarios, respectively, depending on the given machine learning algorithm and the type of document encoding.

Table 5. Machine Learning Algorithms for DDO

<table>
<tr><th rowspan="2">Document Encoding</th><th rowspan="2">Three-phases-frame</th><th colspan="2">Four-phases-frame</th></tr>
<tr><th>Text Clustering</th><th>Text Categorization</th></tr>
<tr><td rowspan="2">Numerical Vectors</td><td rowspan="2">Kohonen Networks</td><td>Kohonen Networks</td><td>SVM</td></tr>
<tr><td>Kohonen Networks</td><td>Back Propagation</td></tr>
<tr><td>Bags of Words</td><td rowspan="3">Single Pass Algorithm</td><td>Single Pass Algorithm</td><td>K Nearest Neighbor</td></tr>
<tr><td>Numerical Vectors</td><td rowspan="2">Single Pass Algorithm</td><td rowspan="2">Naïve Bayes</td></tr>
<tr><td>String Vectors</td></tr>
<tr><td>String Vectors</td><td>NTSO</td><td>NTSO</td><td>NTC</td></tr>
</table>

3.2 Components of DDO System

The implementation of DDO systems involves the three components: text clustering, cluster identification, and text categorization. DDO systems are executed by the coordination of the three components. Their coordination will be described in section 3.3. In this section, the first subsection defines the concept of text clustering and describes its role for implementing DDO systems, the second subsection deals with process of cluster identification for browsing, and the third subsection covers the role of text categorization corresponding to the phases, 'classifier training' and 'document classification'..

3.2.1 Text Clustering

This subsection describes the concept and the roles of text clustering for DDO implementation. Text clustering is involved in building an organization of documents, together with cluster identification which is described in subsection 3.2.2. The roles of text clustering for DDO implementation are initial organization and reorganization. The strategies for reorganization are batch reorganization and interactive reorganization. Therefore, this subsection describes initial organization, batch reorganization, and interactive reorganization as well as the concept of text clustering.

Figure 18 illustrates the generic concept of text clustering. As defined in section 1.1, text clustering is the process of partitioning a particular collection of documents, given as its input and presented in the left side of figure 18, into sub-collections, called clusters, of similar documents in contents, given as its output presented in the right side of figure 18. The clusters are not labeled, yet.

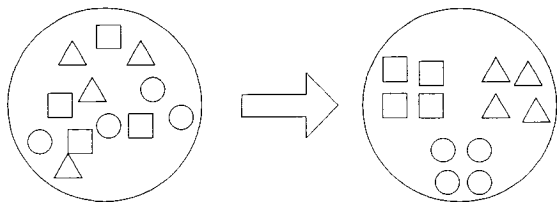

Figure 18. The concept of text clustering

One of the roles of text clustering for the DDO implementation is its initial organization, which is defined as the process of building an organization of documents which were not previously organized. When a particular textual information system is opened initially, it has no documents at all. Afterward, users or administrators add documents to the system continuously. If the system has a sufficient number of documents, their organization is started using the techniques of text clustering. The condition for starting the initial organization is given as a parameter to the system. It consists of a number of documents necessary for the process to start.

The other role of text clustering is reorganization, which is defined as the process of building an organization of documents which were previously organized. One of its strategies is batch reorganization. In this strategy, an organization of documents is built without considering their previous organization. Batch reorganization is identical to initial organization, since its direct input is a collection of all of given documents. Although some documents are deleted from a particular textual information system, the total number of documents in the system increases continually as time goes, because the number of added documents is larger than that of deleted ones, in general. Batch reorganization is costly time wise, when the number of documents increases. This is because the computations involved in text clustering are quadratic in the number of documents.

The other type of reorganization is interactive reorganization, which means the process of building an organization of documents taking into consideration their previous organization; this strategy consists of modifying the previous organization of documents, cluster by cluster. For each cluster, if its size is dominantly large, it is partitioned into several smaller clusters, using techniques of text clustering. If its size is dominantly small, it is merged with another cluster. This is because a sparse cluster can not provide a

sufficient of sample documents to train classifiers involved in maintenance mode of DDO system. It is desirable to merge such a cluster with its most similar cluster in contents. The conditions for partitioning or merging clusters are set as two parameters: minimum cluster size and maximum cluster size. These two parameters will be explained in the next paragraph. The strategy for, interactive reorganization, is more efficient than batch reorganization, since text clustering is not performed for all documents, only documents in each the clusters. In this strategy, the documents in clusters with their medium size, between these two parameters, are not involved in reorganization. However, this strategy is more complicated to implement and apply than batch reorganization, since it requires an optimal configuration of these two parameters involved in partitioning and merging clusters.

The parameters involved in text clustering for DDO implementation are listed as follows.

- **initial collection size**: *the number of documents necessary to initiate the organization of documents*
- **the number of clusters**: *the number of clusters decided as the result of text clustering where using the Kohonen Networks and NTSO approaches*
- **critical similarity**: *the similarity between two documents determining whether a document shall be included in one of existing clusters or in a newly created cluster where using the single pass algorithm as an approach to text clustering*
- **minimum cluster size**: *the size of a cluster used to determine whether a cluster shall be merged or not*
- **maximum cluster size**: *the size of a cluster used to determine whether it shall be partitioned or not*

When the number of documents is greater than the parameter, **initial collection size**, their initial organization is started. The parameter, **critical similarity**, was explained in detail in subsection 2.2.2. In the strategy, interactive reorganization, if a cluster includes more documents than the parameter, **maximum cluster size**, it is partitioned into several smaller clusters, and if a cluster includes fewer documents than the parameter, **minimum cluster size**, it is merged with another cluster.

3.2.2 Cluster Identification

The first phase of DDO implementation, text clustering, was discussed in subsection 3.2.1, while this subsection discusses the second phase, cluster identification. In this phase, clusters of documents resulting from text clustering are labeled. Although this phase is regarded as a trivial task in previous research on text clustering, its significance is to provide the access to the document collection by browsing. This subsection will describe the conditions and the process of cluster identification.

Both text clustering and cluster identification are involved in creating an organization of documents; these two phases automate the preliminary tasks for text categorization. The process of cluster identification should satisfy three following conditions to generate labeled documents suitable for text categorization, the next phase to this phase in DDO implementation.

- *The name of each cluster should be unique*

  More than two clusters are not allowed to have their same name. In other words, the documents belonging to different topics should not be labeled with same name.

- *The name of each cluster should be relevant to the contents of its documents.*
  The name of each cluster should imply the contents of its documents. For example, the word, "sports" is more suitable to the cluster of news articles about Olympics than the word, "society". Labeling clusters with their unique numbers [Hatzivassiloglou, et al. 2000] satisfies the first condition but violates this condition. Assigning numbers to clusters as their identifiers satisfies the first condition, but violates this condition.
- *The name of each cluster should consists of one or a couple of words*
  If the names of clusters are too long [Wu, et al. 2001], they are not suitable as the labels of documents and lead to the difficulty for browsing them.

Figure 19 illustrates the process of cluster identification. A list of unnamed clusters where each cluster includes documents similar in contents as the result of text clustering is given as the input of the process. All documents were indexed into a list of words, during the preprocessing phase. The matrix, Γ, word by cluster, is built as follows,

$$\Gamma = \begin{bmatrix} \varphi'_{11} & \varphi'_{12} & \cdots & \varphi'_{1|C|} \\ \varphi'_{21} & \varphi'_{22} & \cdots & \varphi'_{2|C|} \\ \cdots & \cdots & \cdots & \cdots \\ \varphi'_{N1} & \varphi'_{N2} & \cdots & \varphi'_{N|C|} \end{bmatrix}$$

where each column and each row indicate a cluster and a word respectively. Each of its elements, φ'_{ij}, indicates CNI (Cluster Name Index) and is computed with equation (30),

$$\varphi'_{ij} = CNI_j(w_i) = \frac{\varphi_j(w_i)}{\sum_{k=1}^{|C|} \varphi_k(w_k)} \quad (30)$$

where $\varphi_j(w_i)$ is computed with equation (31),

$$\varphi_j(w_k) = \sum_{k}^{|c_j|} \phi_k(w_i) \quad (31)$$

where $\phi_k(w_i)$ $1 \le k \le |c_j|$ indicates the presence, the frequency, or the weight of the word, w_i in the document, d_k, as discussed in section 2.1. In other words, the function, $\phi_k(w_i)$ indicates the degree of the importance of the word, w_i, in the document, d_k. The function $\varphi_j(w_i)$ is the degree of its importance in the cluster, c_j: the summation of its importance of all documents included in the cluster, c_j. Hence, equation (30) implies that CNI indicates the relative importance of a word in the given cluster with respect to the whole corpus. The word corresponding to the highest CNI, $\varphi'_{\max j} = \max_i \varphi'_{ij} = \max_i CNI_j(w_i)$, which indicates the maximum in the column of the matrix, Γ, corresponding to the cluster, c_j is assigned to it, as its name. This process, illustrated in figure 19, generates the list of named clusters as its output. Given n words and m clusters, the complexity of cluster identification using the algorithm illustrated in figure 19 becomes $O(nm)$.

Input and parameters: a list of unnamed clusters including similar documents in their contents
Step 1: Index the documents of all clusters into a list of words
Step 2: For each word
Step 2-1: For each cluster
Step 2-1-1: compute each element of the word by cluster matrix with the equation
Step 3: For each cluster
Step 3-1: Select the word with the highest cluster naming index as its name
Output: a list of named clusters

Figure 19. The process of cluster identification

The result of this process should satisfy all of the above conditions. This strategy for cluster identification focuses on the second condition and the third condition. For the first condition, if the current cluster has the same name as that of a previous cluster, this problem is solved by concatenating the word with the next highest CNI to its previous name into its new name. Note that the result of cluster identification depends on the result of text clustering. For example, if documents belonging to the same topic are included in different clusters, these clusters may have the same name with high probability; this may violate the first condition. If documents belonging to different topics are included in the same cluster, the name is not relevant; this violates the second condition.

Although cluster identification is similar as key phrase extraction at appearance, they are different with respect to their goals. The goal of cluster identification is to assign identifiers satisfying the above conditions to clusters, while the goal of key phrase extraction is to obtain key phrases from a document or documents. Techniques of extracting key phrases may be used as means of cluster identification. Note that all of extracted key phrases can not become cluster identifiers. Therefore, it is important for cluster identification to select one or two key phrases as a cluster identifier, after extracting them.

3.2.3 Text Categorization

This section describes the role of the two phases, classifier training and document classification for DDO implementation using both scenarios. The role of these two phases is to maintain the organization of existing documents added or deleted after the initial document organization was completed. In the three-phases-scenario, only document classification is involved, while in the four-phases-scenario, both of them are involved. This section describes the process of maintaining the organization of documents, when documents are successively added and deleted both scenarios.

Figure 20 illustrates that text categorization consists of classifier training and document classification. In classifier training, the parameters of a classifier are optimized using the given sample labeled documents. In document classification, an unseen and unlabeled document is classified using the optimized parameters. In the four-phases-scenario of DDO implementation, the documents included in their organization built in text clustering and cluster identification correspond to sample labeled documents involved in classifier training, while successively added documents correspond to unseen and unlabeled documents involved in document classification.

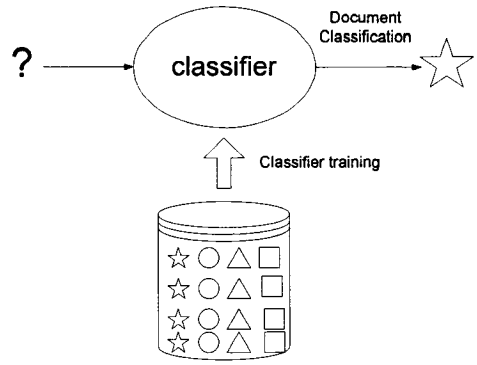

Figure 20. The process of text categorization

The output of text clustering is not only a list of clusters but also a list of their corresponding prototypes. In single pass algorithm, the first document of each cluster is its prototype. Both, Kohonen Networks and NTSO, generate optimized prototypes of clusters, which are represented as numerical vectors and string vectors, respectively. For each added document, its similarity with each prototype is computed, and it is arranged into the cluster corresponding to the prototype with highest similarity, in the document classification phase. The process of computing such similarity was already described in section 2.1.

Since document classification in the three-phases-scenario is performed based on the similarity between each prototype and each document, each cluster is implicitly approximated as the hyper sphere centered at the prototype. In reality, each cluster in the organization has an irregular shape, not a hyper sphere. Since all documents contained in the organization are involved in optimizing the parameters of a classifier, in the four-phases-scenario, it is implied that each cluster will have an irregular shape. Therefore, a more robust document classification is expected in the four-phases-scenario than in the three-phases-scenario. The problem of the four-phases-scenario is that the maintenance of the organization to successively added documents is delayed until classifier training is finished. Hence, there is a trade off between two scenarios of DDO implementation.

3.3 Cycle of Creation Mode and Maintenance Mode

This section describes the cycle of the creation mode and the maintenance mode in DDO systems as a coordination of the three components described in the previous section. As users continue to add and delete documents, the content of organized documents changes gradually. Therefore, these documents should be reorganized, when the current content is sufficiently different from the original content in the organization. This reorganization indicates the transition from document classification to text clustering in both scenarios of DDO implementation. Hence, this section describes three heuristic methods for determining the transition.

Before discussing the three heuristic methods, the terms involved in describing them should be defined. A *transaction* is defined as the action of adding one or several documents or deleting one or several documents. The *original distribution* means the distribution of documents over clusters, just after the transition from the 'creation' mode to the 'maintenance' mode. The *current distribution* means the distribution of documents over clusters when a transaction is performed to the organization, during the 'maintenance' mode.

The determination of the transition depends on the difference between the original distribution and the current distribution, when a transaction is achieved on the organization. If the degree of this difference is higher than the critical value given as a parameter, the reorganization of documents is decided. Based on this difference, three heuristic methods are presented as follows.

- *Transaction based Method*
- *Average Cluster Size based Method*
- *Average Cluster Difference based Method*

In the transaction based method, when a transaction is performed on the organization, the reorganization of documents is started, if the current number of transactions is higher than the critical value given as a parameter. Its advantage is that it is the simplest of the three methods. Its disadvantage is that an unnecessary reorganization may happen; the given documents may be reorganized in spite of little change in their content. If the deletion and the addition of the same documents occur sequentially, the number of transactions is increased without any changes to the organization's contents. This may lead to an unnecessary reorganization.

In the average cluster size based method, when a transaction is performed on the organization, the reorganization of documents is started, if the difference in the average size of clusters between the current distribution and the original distribution is higher than the critical value. This method addresses the disadvantage of the first method; even if users delete and add the same documents, the average size of clusters is not changed. But its disadvantage is that a necessary reorganization may be missed; if the addition of additional documents and the deletion of other documents occur alternatively, the average size of the clusters does not change much. Although the content of the existing organization is changed, the reorganization does not occur in this method.

In the average cluster difference based method, when a transaction is performed on the organization, the reorganization of documents is started, if the difference in the list of document identifiers between the current distribution and the original distribution is higher than the critical value. The difference indicates the average number of document identifiers which is included in the list of either of the original distribution or the current distribution, not both of them, over clusters. This method solves the disadvantages of the above two strategies. Even if the deletion and the addition of the same documents occur, the difference is not increased, since the identifiers of these documents are included in the list of both distributions. Therefore, this method prevents an unnecessary reorganization. While this method solves these two problems, it is very expensive; it requires keeping the record of document identifiers in the original distribution, until their reorganization and comparing the list of the original distribution with that of the current distribution, whenever a transaction occurs.

When we use the k mean algorithm, Kohonen Networks, or NTSO, as an approach to the component of text clustering in DDO systems, we need to consider the number of clusters whenever the status moves back to the creation mode, since in the clustering algorithms the number of clusters is given as their parameter. We can address this problem using one of two ways. One is a simple way, where a number of clusters is generated automatically by dividing the total number of documents by a particular constant. The constant is fixed initially and given as a parameter of DDO system. The other way is the complicated method, where the number of clusters is determined based

on the number of documents which were deleted and added during the maintenance mode. A fixed integer is given as the initial number of clusters. If many documents are added, the number of clusters in the next creation mode should be increased. If many documents are deleted, then, on contrary, it should be decreased.

Chapter 4 A Better Document Representation for DDO: String Vectors

The previous chapter discussed concepts, components, and their coordination involved in implementing DDO systems. This chapter discusses another representation of documents, called string vector, for implementing DDO systems. It is composed of four sections. The first section defines string vectors and discusses their advantages as a better representation of documents than the two existing ones discussed in section 2.1. The second and third sections discuss the unsupervised and supervised neural networks that use string vectors as the document representation for text clustering and text categorization. The last section presents the results of comparing these neural networks with the traditional approaches to text clustering and text categorization described in chapter 2.

4.1 Description of String Vectors

This section introduces an alternative representation of documents to the traditional ones described in section 2.1. It first defines the representation in detail, and it then discusses its advantages over the traditional representations described in section 2.1, for encoding documents. This section also introduces a similarity matrix necessary for computing string vectors, and discusses its properties mathematically. Since string vectors and bags of words are similar to each other in their appearance, their differences are illustrated using examples.

T. Jo proposed a new supervised neural network, called NTC (Neural Text Categorizer), that uses string vectors as its input vectors, for text categorization in 2000 [Jo 2000]. In 2004, he validated its performance by comparing it with the main traditional approaches to text categorization, such as Naïve Bayes, SVM (Support Vector Machine), and Back Propagation on Reuter 21578, as a standard test bed [Jo 2004]. His study [Jo 2004] showed that a string vector represents a document better than a numerical vector, in text categorization. In 2005, T. Jo proposed a new unsupervised neural network, called NTSO (Neural Text Self Organizer), that uses string vectors as its input vectors and its weight vectors, and validated it by comparing it with Kohonen Networks with respect to their performance and their speed of clustering [Jo and Japkowicz 2005].

A string vector is defined as an ordered set of words with fixed size, independent of the length of the given document. The string vector representing the document, d_i is denoted by $d_i^s = [w_{i1} \quad w_{i2} \quad \ldots \quad w_{in}]$, where n is the dimension of the string vector, d_i^s. An arbitrary example of a size 4 string vector is expressed as [computer, software, hardware, machine].

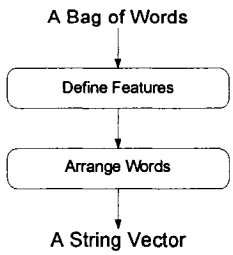

Figure 21. The process of mapping a bag of words into a string vector

Here is the process by which a string vector is created. From the given document, a bag of words denoted by $d_i^b = [w_{i1} \quad w_{i2} \quad \ldots \quad w_{il}]$ is generated, as an intermediate representation for a string vector, through the process illustrated in figure 3 in chapter 2. Figure 21 illustrates the process of mapping a bag of words into a string vector. The dimension of a string vector is determined and the properties of words, such as their frequency, correspondence to the document, their weight as a degree of their content importance, are defined as the features of that vector. For example, if its dimension is set to four, the features' nature may be defined as the following properties: the word with the highest weight, a random word positioned in the title, a random word positioned in the first sentence, and a random word positioned in the last sentence. For simplicity and convenience of implementing the automatic process of encoding documents into string vectors, we defined the properties of string vectors as the highest frequent word, the second highest frequent word, the third highest frequent word, and the fourth highest frequency word in the document. In general, the dimension of string vectors is smaller than the size of bags of words[9]. To each property given as a feature, its corresponding word is arranged so as to build a string vector.

For example, a document is given as follows.

The Labor Department reported an increase of 138,000 jobs last month, led by a rare spurt of 19,000 new manufacturing hires, and more solid gains in construction, finance, health care, professional and technical work.

But the number of new jobs created in the previous two months was revised down by 36,000, and retailers pared their work forces, apparently in anticipation of more subdued consumer spending.

Wage growth surged by 0.5 percent, for a 3.8 percent gain in the last year -- the largest since August 2001. But with unemployment steadying out at 4.7 percent, the picture of moderating growth soothed financial markets, which have worried that more robust growth like the first quarter's 4.8 percent pace would spark inflation and prompt the Federal Reserve to keep raising interest rates.

The Dow soared by 139 points to its third-highest close, 11,578, a level not seen since January 2000. The Standard & Poor's 500 index, another blue-chip stock gauge, climbed to its highest level in five years at 1,326. But the technology-led Nasdaq Composite Index, which skyrocketed during the Internet bubble, remains at less than half its previous high.

The stock market's gains this year have come on signs that the economy is maturing into a stage of expansion in which workers start to partake more in the profits corporations are enjoying even as growth settles down to a steady trot.

"The low unemployment rate of 4.7 percent suggests we have very strong labor demand, and strong labor demand usually is followed by wage increases," said Edward P. Lazear, newly appointed chairman of the White House Council of Economic Advisers.

Less of the earnings from improved output or productivity by workers will be devoted to profits in the future and more to wages, he said. "We are moving into the phase where we expect that wage growth will catch up and take over productivity growth."

[9] The size of a bag of words is from 100 to 300, depending on the size of the given document. But the dimension of a string vector is set less than 50 [Jo 2004][Jo and Japkowicz 2005][Jo 2000].

We can encode the above document into a bag of words, a numerical vector, a string vector. A bag of word representing the above document is {labor, department, increase, job, month, ...}. Let's assume that five features selected from a corpus are 'labor', 'job', 'profit', 'bubble', and 'index'. Therefore, the above document is encoded into a five dimensional numerical vector, [0 1 1 1 1], if attribute values are defined as binary values indicating absence or presence of the features. Let's assume that we define features for string vectors, as a first non-stop word of each of the seven paragraphs; we encode the above document into seven dimensional string vector, [labor, number, wage, dow, stock, low, earning]. However, in this research, we defined features for string vectors in the descending order of frequencies of words for easy and simple implementation.

String vectors have their own advantages over the two traditional types of document encodings mentioned in section 2.1, in terms of processing documents. First, an advantage of string vectors over bags of words is that string vectors have lower complexity in computing their similarities than bags of words. For bags of words, their similarities are computed based on complete relations (shown in the right of figure 22) of their elements, while similarities of string vectors are computed based one to one relations (shown in the left of figure 22) of their elements. This will be discussed formally at the end of the section. Therefore, if each string vector or bag of words contains n elements, string vectors have the linear complexity of their elements, $O(n)$, for computing their similarities, while bags of words have the quadratic complexity, $O(n^2)$, for doing that.

Second, an advantage of string vectors over numerical vectors is that problems caused by sparse distributions of numerical vectors can be avoided, since the concept of sparse distribution exists only in domain of numerical values. When documents are encoded into numerical vectors, most of them may have their sparse distribution, where zero values are dominant, because some of words over a collection of documents are selected as features of numerical vectors. In sparse numerical vectors, the probability that values of their inner products are zeros is very high; almost all of their similarities also have zero values, since similarities of numerical vectors are computed based on their inner products as shown in equation (5), (6), and (7) of section 2.1. Especially if the single pass algorithm or k nearest neighbor in particular is adopted as an approach to text clustering or text classification, sparse numerical vectors lead to very poor robustness for tasks, because almost all of similarities have zero values. In order to mitigate this problem, a given problem of text classification tends to be decomposed into multiple binary classification problems, when numerical vectors are adopted as representations of documents [Sebastiani, 2002]. If we adopt string vectors for encoding documents, instead of numerical vectors, we do not need the decomposition.

Before performing operations on string vectors, from a particular corpus, we need to define a similarity matrix, a word by word matrix whose entries indicate similarities of all possible pairs of words. The idea of building a similarity matrix is based on the research of [Cristianini et al 2000]. In research, a similarity matrix is also built, in order to define a kernel function for employing SVM. By encoding documents into numerical vectors, a term by document matrix whose rows are numerical vectors representing documents is built from a corpus. A word by word matrix, which is called term by term matrix in the literature [Cristianini et al 2000], is induced by performing a product to its transpose and the matrix, itself. But there are two drawbacks in the matrix. One is that each entry

indicating a similarity of a pair of words is not a normalized value; the maximum of similarities is not defined explicitly. The other is that a similarity of a word to itself varies depending on its frequency and distribution over documents; a similarity of two different words may have its higher value than that of two same words.

In 2000, Cristianini et al proposed LSI (Latent Semantic Index) which is built by multiplying a word by document matrix with a document by word matrix from a corpus [Cristiani 2001]. In 2001, Turney proposed another semantic similarity measure between two words, called 'pointwise mutual information', and compared his proposed measure with LSI in information retrieval concerning TOEFL [Turney 2001]. Although the two works, like ours, represent ways of building a similarity matrix, this research makes a different definition of semantic similarity between two words for two reasons. The first reason is that their proposed measure does not provide a similarity measure as a normalized value; we can not judge the relative strength of the given similarity without knowing the maximum. The second reason is that a semantic similarity between two identical words is dependent on the given words; the semantic similarity could, in certain cases, not be maximal.

To build a similarity matrix, we must index documents into lists of words contained in a given corpus and union the lists. The process of indexing them was already illustrated in figure 3. Note that stop words were removed in this process[10].

In this research, each entry has a normalized value ranging from zero to one, and each diagonal entry indicating the similarity between a word and itself always has the value, 1.0 the maximum of semantic similarities. In the word by word matrix of this research there are words denoted by $w_1, w_2, ..., w_N$, where N is the number of unique words in the given corpus, except for stop words. It is denoted by the matrix, N by N,

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

and a simple example of it is illustrated in table 6, where N is eight. Its elements are denoted by $s_{ij}(1 \le i, j \le N)$, and defined as a semantic similarity, $s_{ij} = sim(w_i, w_j)$, between two words, w_i and w_j. Each element, $s_{ij}(1 \le i, j \le N)$, is computed based on the collocation of two words, w_i and w_j, within same document, using equation (32),

$$s_{ij} = sim(w_i, w_j) = \frac{\sum_{d_r \in D_i \cap D_j} (\phi_r(w_i) + \phi_r(w_j))}{\sum_{d_p \in D_i} \phi_p(w_i) + \sum_{d_q \in D_j} \phi_q(w_j)} \quad (32)$$

where D_i is the set of documents including the word, w_i, and D_j is the set of documents including the word, w_j, and $\phi_o(\cdot)$ is the function of a word, unique to a particular document, d_o. The function, $\phi_o(\cdot)$ of a word, means any information about the word in

[10] We built a list of stop words as a file, and the indexing program removes stop words included in a document or documents with the reference to the file.

the document, d_o; it indicates binary value indicating its presence, its absolute or relative frequency, its weights using equation (4), or others. Equation (32) indicates the ratio of documents including both words, to documents include either of them with respect to their occurrences, frequencies, and weights. Note that equation (32) is different from equation (1), (2), and (3). The goal of equation (32) is to compute a similarity between two words, while the goal of equations (1) (2), and (3) is to compute a similarity between two documents encoded into bags of words. Therefore, in equation (32), the portion of documents including both words is the basis of computing a similarity between two words, while in equation (1) (2), and (3), the portion of words included in both documents is the basis of computing a similarity between two documents. This function was already described in section, 3.2.2. The similarity matrix has the two following properties, shared with all string vectors.

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

Since two words at position, i are a word and itself, the column and the row are identical to each other and the similarity as computed with equation (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. $s_{ij} = s_{ji} (1 \le i, j \le N)$

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>

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

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

In the process of building a similarity matrix from a corpus, the input is a corpus containing documents and the output is the matrix with the two properties. The documents are indexed into lists of words, and the lists are unified into a list. The process

of indexing them was already described in section 2.1.1, and stop words were removed. After that, we define a function, $\phi_o(\cdot)$, by selecting an information among binary value indicating its presence, its absolute or relative frequency, its weights using equation (4), or others. We build a matrix by computing semantic similarities of all pairs of words using equation (32). When N words are given, the time for building a similarity becomes $\frac{1}{2}N(N-1)$; the time complexity becomes $O(N^2)$. Therefore, the process of building a similarity matrix is very expensive. However, we build it not frequently, since we can reuse continually the similarity matrix to another corpus, unless the current domain is much different from the domain of the corpus from which the similarity matrix was built.

Table 6. A simple example of a similarity matrix

	Computer	Software	Hardware	Machine	Automobile	Bus	Car	Airplane
Computer	1.0	0.8	0.7	0.5	0.2	0	0	0.2
Software	0.8	1.0	0.2	0.3	0.1	0	0	0.1
Hardware	0.7	0.2	1.0	0.9	0.8	0.5	0.6	0.6
Machine	0.5	0.3	0.9	1.0	0.7	0.6	0.7	0.8
Automobile	0.2	0.1	0.8	0.7	1.0	0.9	0.9	0.6
Bus	0	0	0.5	0.6	0.9	1.0	0.8	0.4
Car	0	0	0.6	0.7	0.9	0.8	1.0	0.3
Airplane	0.2	0.1	0.6	0.8	0.6	0.4	0.3	1.0

If a similarity matrix is built from a corpus by computing similarities between all possible pairs of words using equation (32), we can compute the similarity between two string vectors, denoted by $d_i^s = [w_{i1}, w_{i2}, ..., w_{in}]$ and $d_j^s = [w_{j1}, w_{j2}, ..., w_{jn}]$. The similarity $sim(w_{ik}, w_{jk})$ between two words, w_{ik} and w_{jk} is obtained by looking up the entry with the row corresponding to the word, w_{ik} and the column corresponding to the word, w_{jk} or with its reverse, from the similarity matrix. The similarity between two string vectors d_i^s and d_j^s is computed using equation (33).

$$sim(d_i, d_j) \approx sim(d_i^s, d_j^s) = \frac{1}{n}\sum_{k=1}^{n} sim(w_{ik}, w_{jk}) \quad (33)$$

As illustrated in equation (33), the similarity between two string vectors is the average over the similarities of their elements.

We illustrate the process of computing the similarity between two string vectors representing documents through a simple example. The similarity matrix is given as table 6, and two string vectors are given as [computer, software, hardware, machine] and [automobile, hardware, machine, airplane]. The 1:1 links presented in the left side of figure 22 is involved in computing the similarity between these two string vectors. We obtain four following similarities by looking them up in table 6.

- The similarity between "computer" and "automobile"→0.2
- The similarity between "software" and "hardware"→0.2
- The similarity between "hardware" and "machine"→0.9
- The similarity between "machine" and "airplane"→0.8

Therefore, the similarity between these two string vectors is the average over them, 0.525.

Equation (32) implies that a similarity between two words is evaluated differently depending on the documents in the corpus for building a similarity matrix. If two semantically irrelevant words occur very frequently in the same document, their similarity is evaluated as high. For example, although two words, "dog" and "flower", are irrelevant semantically, if the two words occur frequently in their same documents included in the corpus, the similarity between the two words is estimated as a high value (a value close to one). In collection of documents on biology, the two words, "dog" and "flower" collocate less frequently, since one belongs to plant, and the other belongs to animal. They have very different semantic similarity between them. When string vectors are used as a method of representing documents, more consideration of the domain of the corpus for building a similarity matrix is necessary than other methods. In other words, especially when documents are encoded into string vectors, we must not use formal documents such as project proposals, patent abstracts, and research papers, as the corpus for building a similarity matrix for processing informal documents such as novels, short stories for children, and email.

Note that a string vector is fundamentally different from a bag of words with respect to their similarity computations, although both of them have the same elements. As discussed above, a string vector is defined as an ordered set with a fixed size independent of the size of the given document, while a bag of words is defined as an unordered set with a flexible size depending on the size of the given document. Given a similarity matrix built from a corpus, the similarity between the two bags of words, $d_i^b = \{w_{i1}, w_{i2}, ..., w_{il}\}$ and $d_j^b = \{w_{j1}, w_{j2}, ..., w_{jm}\}$ is computed using equation (34).

$$sim(d_i, d_j) \approx sim(d_i^b, d_j^b) = \frac{1}{lm} \sum_{k=1}^{l} \sum_{h=1}^{m} sim(w_{ik}, w_{jh}) \quad (34)$$

As illustrated in equation (34), all links between them are considered to compute their similarity, as illustrated in the right side of figure 22. Therefore, the complexity of equation (34) is $O(lm)$, while that of equation (33) is $O(n)$. In computing the similarity between two documents, a string vector has much lower complexity than a bag of words.

We show that a bag of words is different from a string vector by computing a similarity between the two bags of words with the same elements as those of the above two string vectors, given as {computer, software, hardware, machine} and {automobile, hardware, machine, airplane}. The sixty links, all possible pairs between these two bags of words presented in the right side of figure 22, are considered to compute their similarity, unlike the two string vectors. If the similarities corresponding to these sixty links are looked up in table 6 and averaged, the similarity between these two bags of words is 0.606. Hence, this example shows that a bag of words is a document encoding different from a string vector in spite of their same elements.

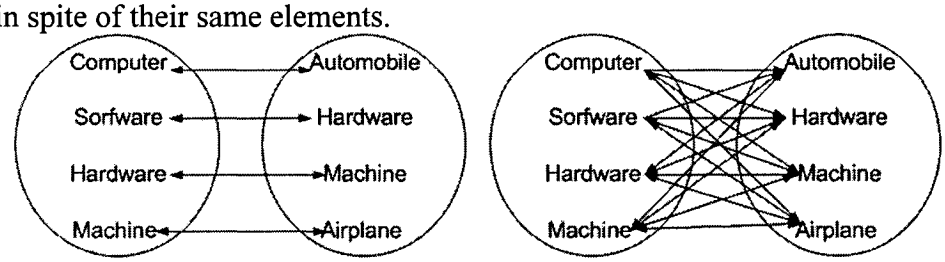

Figure 22. 1:1 links of string vectors in the left and complete links of bags of words in the right

Table 7 summarizes the different properties underlying the three document encodings described in this section. The properties of a bag of words and a string vector corresponding to the second row, the third row, the fourth row, and the fifth row were already discussed in this subsection. The properties of a numerical vector corresponding to the second row, the third row, and the fourth row were already discussed in section 2.1 and this section. The operations of numerical vectors, such as their inner product and the Euclidean distance between them, are performed with 1:1 links between them.

Table 7. The comparison of the three document encodings

	A bag of words	A numerical vector	A string vector
The nature of a set	Unordered set with its various size	Ordered set with its fixed size	Ordered set with its fixed size
Features	Not Defined	Words	Natures of Words
Elements (Feature Values)	Words	Numerical values	Words
Links for similarity computation	Complete link	1:1 link	1:1 link

String vectors are independent of a given corpus in their representation. The features defined in this research are based on only frequencies in the given document. A corpus does not influence on representing documents into string vectors. However, note that operations on string vectors are dependent on a corpus, since we must build a similarity matrix from a corpus. The operations are applicable to NTSO, not to NTC. Therefore, using string vectors for NTSO is influenced by a corpus, but using them for NTC is not so.

4.2 Text Clustering with String Vectors: NTSO

The novel approach we introduce for text clustering is NTSO (Neural Text Self Organizer) which is an unsupervised neural network using string vectors as its input vectors and weight vectors. Its architecture is identical to that of Kohonen Networks, as illustrated in figure 5 of section 2.2.2.2, and was already discussed in the description of Kohonen Networks. The process of clustering documents with NTSO is almost identical to that of clustering them with Kohonen Networks, except for the processing of the string vectors. In the description of the process of clustering documents with this approach, only its differing points will be explained.

Before discussing the learning algorithm of NTSO (Neural Text Self Organizer), let us discuss the concept of *inter-words*. Inter-words are words whose similarities to two words are greater than the similarity of those words. Such similarity is defined by the similarity matrix built from the given corpus. In order to find inter-words between two words, we first find the similarity between these two words from the similarity matrix. We then, extract words with higher similarity to either of them from the similarity matrix. The set of inter-words between two words denoted by w_i and w_j, denoted by I_{ij} is expressed by equation (35).

$$I_{ij} = I(w_i, w_j) = \{w_k \mid w_k \quad s(w_i, w_k) \geq s(w_i, w_j) \wedge s(w_j, w_k) \geq s(w_j, w_i)\} \quad (35)$$

Figure 23 shows the process of clustering documents using NTSO. A learning rate is not included in NTSO as its parameter, since its learning involves the computation of string

vectors, and not numerical vectors. In this approach, the given documents are encoded into string vectors, which are different from bags of words. At first, the given weight vectors are initialized with one of the two following strategies.

- **Sample based Initialization**: Initializes the weight vectors by selecting some of the given training examples.
- **Corpus based Initialization**: Initializes the weight vectors by selecting words from the corpus as their elements, at random

Whenever each input vector is given, the similarities between the weight vectors and the input vector, are computed using equation (33) given in section 4.1. Only the weight vector of the competitive node with highest similarity is updated by replacing each of its elements with an inter-word selected randomly from the set, according to equation (35), between each of its elements and those of an input vector. Let's denote a random element of a particular set, A by rand(A). Therefore, a random element of inter-word set, I_{ij} between two words, w_i and w_j is denoted by rand(I_{ij}). The rules for updating the weight vectors from the weight vector, $weight_{\max}$, and the input vector, $input_j$, respectively are given in equation (36),

$$\begin{aligned} &weight_{\max}(t-1) = [w_{\max 1}, w_{\max 2}, \ldots, w_{\max n}] \\ &input_j = [w_{j1}, w_{j2}, \ldots, w_{jn}] \\ &wwight_{\max}(t) \rightarrow [\text{rand}(I(w_{\max 1}, w_{j1})), \text{rand}(I(w_{\max 2}, w_{j2})), \ldots, \text{rand}(I(w_{\max n}, w_{jn}))] \end{aligned} \quad (36)$$

where $weight_{\max}$ is the weight vector corresponding to the competitive node with the highest value, $input_j$ is the jth input vector, and rand($I(w_{ik}, w_{jk})$) is a random element of the inter-word set between the two words, w_{ik} and w_{jk}. The reason of selecting one of inter-words is that words can not be continuous quantities differently from numerical value, and learning rate can not be applicable. Therefore, in stead of applying learning rate, we adopt random selection of inter words for updating weights. This process of computing the similarity between each input vector and each weight vector and updating the weight vector with the highest similarity is repeated as many times as experimented by the fixed number given as a parameter. Each document is represented into a d dimensional string vector. Given n documents, m clusters, and k iterations, NTSO has complexity for clustering, $O(dkmn)$, which is identical to that of Kohonen Networks.

NTSO may be viewed as another version of Kohonen Networks that uses string vectors, as an alternative strategy for encoding documents. So its advantage over the first approach, single pass algorithm, is the same to that of Kohonen Networks. Its advantages over Kohonen Networks are that the process of clustering documents with this approach is free from the main two problems caused when encoding them into numerical vectors and that it executes text clustering faster than Kohonen Networks, empirically, although its forward complexity is the same as that of Kohonen Networks. The reason is that it clusters documents better than Kohonen Networks using a smaller dimensionality of string vectors and a smaller number of iterations. The comparison between the two algorithms will be presented in subsection 4.4.1.

Input and parameters: a list of documents, #clusters 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
Step 4-1-2: update the weight vector with the highest similarity using equation (36)
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 23. The process of clustering documents using NTSO

4.3 Text Categorization with String Vectors: NTC

This section describes the supervised neural network, called NTC (Neural Text Categorizer), using string vectors as its input vectors with respect to its architecture, initialization, training, and classification. In 2000, it was initially proposed by Jo to address the two main problems in representing documents as numerical vectors. The approach was validated by comparing it with the three main traditional approaches to text categorization, NB, SVM, and BP on the standard test bed, Reuter21578, in 2004 [Jo 2004].

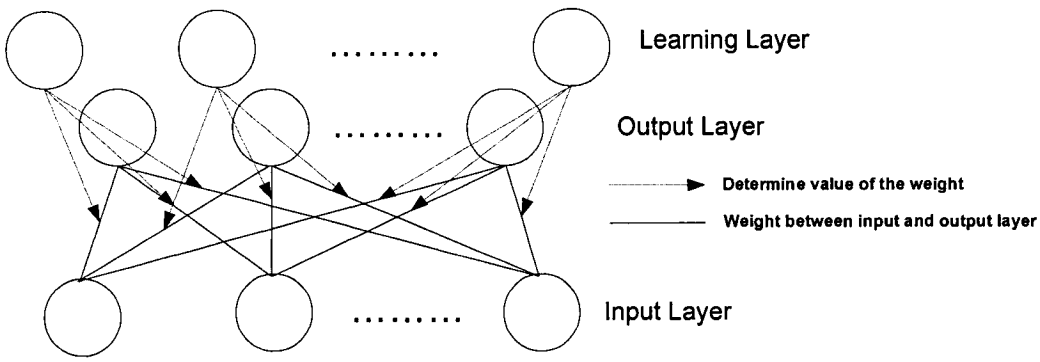

Figure 24. The Architecture of NTC

Figure 24 illustrates the architecture of NTC consisting of three layers: the input layer, the learning layer, and the output layer. The role of the input layer is to receive the input vector given as a string vector; each node corresponds to each of its elements. The learning layer determines the weights between the input layer and the output layer, whenever an input vector is given. The output layer generates the degree of membership of the input vector encoding a document to each category. The conditions for designing the architecture of NTC are given as follows.

- The number of input nodes should be identical to the dimension of the string vectors encoding documents
- The number of learning nodes should be identical to the number of given categories.
- The number of output nodes should be identical to the number of given categories.

Table 8 defines the nodes in these three layers involved in the architecture of NTC. Each input node corresponds to an element of the input vector given as a string vector denoted by $d_k^s = [w_{k1}, w_{k2}, ..., w_{kn}]$, encoding a document, d_i. Each learning node corresponds to a category and is defined as an unordered set of words and their weights denoted by $\mathbf{l}_r = \{(w_{r1}, weight(w_{r1})), (w_{r2}, weight(w_{r2})), ..., (w_{r|\mathbf{l}_r|}, weight(w_{r|\mathbf{l}_r|}))\}$. Given the string vector, d_k^s, the weight between the input node, i_j and the output node, o_r, $weight_r(w_{kj})$, is computed by looking it up in the unordered set expressing the learning node, $\mathbf{l}_r$, as expressed in equation (37). This weight indicates the degree of the membership of the word, w_{kj} in the string vector, d_k^s, in the category, c_r.

$$weight_r(w_{kj}) = \begin{cases} weight(w_{rm}), & \text{if } ((w_{rm}, weight(w_{rm})) \in \mathbf{l}_r) \wedge (w_{kj} = w_{rm}) \\ 0, & \text{otherwise} \end{cases} \quad (37)$$

The value of each output node, o_r is computed using the equation presented in table 8. Therefore, the classified category of the input vector, $\hat{c}_k$ corresponds to the output node with its highest value. Therefore, the process of computing the values of the given output nodes refers to the classification of string vectors encoding documents with the current weights.

Table 8. The Definition of the Input Nodes and the Output Nodes in NTC

	Input Layer	Learning Layer	Output Layer
The Notation of Nodes	$\mathbf{i} = \{i_1, i_2, ..., i_{|\mathbf{i}|}\}$	$\mathbf{l} = \{\mathbf{l}_1, \mathbf{l}_2, ..., \mathbf{l}_{|\mathbf{o}|}\}$	$\mathbf{o} = \{o_i, o_2, ..., o_{|\mathbf{o}|}\}$
The Value of Nodes	$i_j = w_{kj}$	$\mathbf{l}_r = \{(w_{r1}, weight(w_{r1})), ..., (w_{r|\mathbf{l}_r|}, weight(w_{r|\mathbf{l}_r|}))\}$	$o_r = \sum_{j=1}^{|\mathbf{i}|} weight_r(w_{kj})$

In this approach, learning is the process of updating weights connecting the input layer with the output layer to each word, in order to minimize the number of misclassifications. At first, the weights are initialized with the number of the documents including the word within the category. The learning rate, denoted by η, is given as a parameter of this model, like in BP. The value of each output node is computed using the equation of table 8 with the current weights. The target category of the input vector, d_k^s, is denoted by t_k. The weights are updated when the target category, t_k is not consistent with the classified category, $\hat{c}_k$. The output node corresponding to the target category is denoted by o_t, and the output node corresponding to the classified category is denoted by $o_{\hat{c}}$. The rule for updating weights is expressed by equation (38),

$$\text{if } o_t \neq o_{\hat{c}} \begin{cases} weight_t(w_{kj}) \leftarrow weight_t(w_{kj}) + \eta weight_t(w_{kj}) \\ weight_{\hat{c}}(w_{kj}) \leftarrow weight_{\hat{c}}(w_{kj}) - \eta weight_{\hat{c}}(w_{kj}) \end{cases} \quad (38)$$

where $weight_t(w_{kj})$ indicates the weight of the word, w_{kj}, in the target category, given as an element of the string vector, and $weight_{\hat{c}}(w_{kj})$ indicates its weight in the classified category. Equation (38) indicates that the weights of the given input vector are adjusted

by increasing its weights in its target category and reducing those in its misclassified category.

Figure 25 illustrates the process of classifier training and document classification using NTC. In the process of classifier training, the input is a series of the documents included in the organization given as sample documents like other approaches, and the parameters are the learning rate and the number of iterations. In this approach, the features of a string vector should be defined before encoding sample documents and the process of defining such features was already described in section 2.1. The architecture of NTC is designed with the satisfaction of the above three conditions. For each string vector encoding a document, its classified cluster is determined by computing the value of each output node using the equation in table 8. If its classified class is not the same as its target class, its weights, connecting to two output nodes corresponding to its target class and its classified class, are adjusted using equation (38). This process is applied to all the string vectors encoding the sample documents, and is iterated with the fixed number of iterations given as a parameter. After iteration, the process of classifier training generates the optimized weights of the words in each learning node as its output. Each document is represented into a d dimensional string vector. Given n documents and k iterations, NTC has complexity for clustering, $O(dkn)$, which is identical to that of Kohonen Networks.

Classifier Training
Input: A Series of Sample Documents, Learning Rate, and Iteration Number
Step 1: Encode these sample documents into string vectors
Step 2: Design the architecture of NTC
Step 3: Initialize weights in each learning with its document frequency within the corresponding cluster
Step 3: Repeat step 3-1 with the number of iteration
Step 3-1: For each encoded sample document
Step 3-1-1: Compute the values of output nodes of the encoded document with the current weight using the equation in table 9
Step 3-1-2: Update weights to every misclassification using the equation (40)
Output: Optimized weights in each learning node

Document Classification
Input: An unseen document and the optimized weights in each learning node
Step 1: Encode the unseen document into a string data
Step 2: Compute the values of output nodes of the encoded document with the current weight using the equation in table 9
Step 3: Assign the document to the class the document into the class of which the output node has highest value
Output: The class identifier assigned to the document

Figure 25. The Process of Classifier Training and Document Classification using NTC

In the process of document classification, an unseen document and the output of trained classifier are given as its input. The given document is encoded into a string vector. With the optimized weights of each element of the string vector in each learning node given as the input of this process, the value of each output node is computed using the equation in table 8. The document is assigned to the class corresponding to the output node with highest value.

NTC (Neural Text Categorizer) is similar to Perceptron in two respects. First, the value of the output node corresponding to each category is computed using a linear combination of weights between the input layer and the output layer as expressed with

the equation in table 8. Second, such weights are updated only when the target category of each training example is not identical to its classified one. Therefore, this approach has these two properties of Perceptron.

NTC may be confused with back propagation, since both of them have the three layers. However, NTC is more clearly different from back propagation than from Perceptron. In their architecture, the hidden layer of back propagation is different from the learning layer of NTC with respect to their role, as the layer given additionally to the input layer and the output layer. The hidden layer makes the boundary quadratic to nonlinear separable distribution of training examples, while the learning layer determines the weights between the input layer and the output layer. In NTC, the input layer is connected directly with the output layer like the Perceptron, while in back propagation, the input layer is connected not with the output layer, but with the hidden layer.

4.4 Experiments with String Vectors

This section concerns the experiments where the different strategies for representing documents are compared with each other in text clustering and text categorization. In text clustering, NTSO which uses string vectors as its weights vectors and input vectors will be compared with the traditional approaches, single pass algorithm, k-means algorithm, and Kohonen Networks, which use numerical vectors. In text categorization, NTC will be compared with K Nearest Neighbor, Naïve Bayes, Support Vector Machine, and Back Propagation. Goals of the experiments concerned in this section are to validate the two neural networks described in section 4.2 and 4.3 and to determine which of approaches should be adopted for implementing DDO systems.

4.4.1 Experiments in Text Clustering

This section concerns a preliminary experiment for evaluating several approaches to text clustering, in order to provide a basis for selecting some of them for implementing DDO systems. The four clustering algorithms, k means algorithm, single pass algorithm, Kohonen Networks, and NTSO, participate in this preliminary experiment. Among them, k means algorithm is a simplified version of the EM algorithm [Mitchell 1997]; it is adopted as a representative version of the EM algorithm for this preliminary experiment. Two test beds where documents are exclusively labeled are used, in order to evaluate the four approaches easily. The goal of this preliminary experiment is to select some of the four approaches for implementing DDO system, by observing their clustering performance and speed.

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. This collection will be explained in detail in section 4.4.2. 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. Although Reuter21578 is used as a standard test bed, it is excluded for evaluating text clustering approaches, since each document has

more than one label in the collection. Here, text clustering is not overlapping clustering, but exclusive clustering.

Table 9 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 [Mostafa and Lam 2000] [Hatzivassiloglou, et al. 2000][Wu, et al. 2001]. 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 [Sebastiani 2002] [Yang 1999] [Kaski et al 1998] [Kohonen et al 2000], 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.

Table 9. Definition of Parameters of the Four Clustering Algorithms

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 Input dimension = 500
	20NewsGroups	
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

In this experiment, the clustering index, which is measure for evaluating clustering performance based on intra-cluster similarity and inter-cluster similarity, adopted as a performance measure for clustering algorithms. This measure verifies that inter-cluster similarities should be high and intra-cluster similarities should be low in clustering objects, and provides a normalized value between zero and one. The higher the inter-

cluster similarities and the lower the inter-cluster similarities, the higher the clustering index is. This performance measure will be explained in great detail in section 6.2.

Figure 26 presents the results of evaluating the four approaches to text clustering with respect to their 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 these 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.

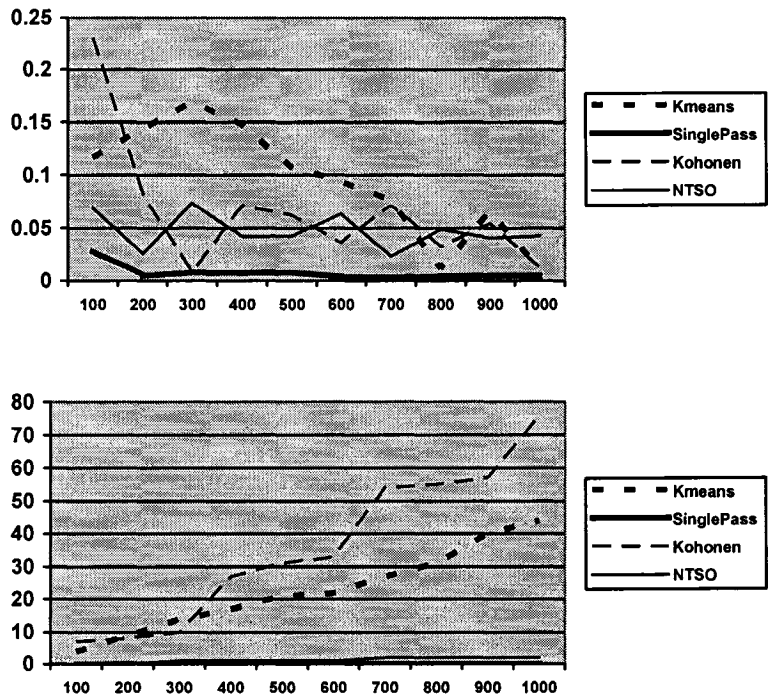

Figure 26. The Results of Evaluating Text Clustering Algorithms on NewsPage.com: Top (Clustering Index) and Bottom (Execution Time)

The top line graph in figure 26 shows that the k means algorithm clusters documents best among the four clustering algorithms from 100 to 700 documents, but the three clustering algorithms except the single pass algorithm are competitive with each other in clustering more than 700 documents. Among the four clustering algorithms, the single pass algorithm clusters documents the poorest, independently of the number of documents, as illustrated in the top line graph of figure 26. The bottom line graph in figure 26 shows that the single pass algorithm and NTSO clusters document with highest speed, together, than the k means algorithm and Kohonen Networks. According to the two line graphs, it is judged that NTSO clusters document with competitive performance with k means algorithm and Kohonen Networks and has a speed competitive with single

pass algorithm. These results imply that NTSO has advantages over the three clustering algorithms considering three factors: clustering performance, clustering speed, and scalability. Therefore, we need to evaluate participated algorithms from 800 documents to 1000 documents. In clustering 1,000 documents, Konhonen Networks and k-means algorithm take 76 seconds and 44 seconds, respectively. The single pass algorithm and NTSO takes less than one second and two seconds, respectively.

Figure 27 illustrates the results of evaluating the four approaches in view of the two factors on the second test bed, '20NewsGroups'. The top line graph presents that Kohonen Networks has the best clustering performance, entirely. It shows also that although the performance of NTSO is not as good as Kohonen Networks, it is competitive with k means algorithm. According to the top line graph and the bottom line graph contained in figure 27, it is judged that NTSO clusters document with its competitive performance and again, with much higher speed, compared with k means algorithm. Considering these two factors, NTSO seems competitive with Kohonen Networks, since NTSO clusters document with higher speed than Kohonen Networks.

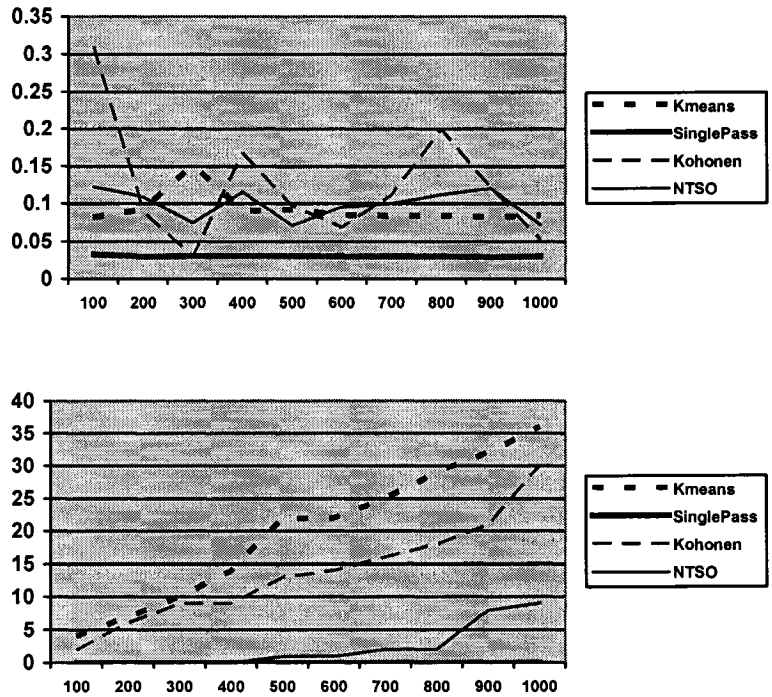

Figure 27. The Results of Evaluating Text Clustering Algorithms on 20 NewsGroups: Top (Clustering Index) and Bottom (Execution Time)

NTSO and Kohonen Networks will thus be adopted for implementing DDO systems. The results illustrated in figure 26 show that NTSO clusters documents with competitive performance to k means algorithm and Kohonen Networks and with competitive speed with single pass algorithm. The results illustrated in figure 26 and 27 shows that Kohonen Networks clusters documents faster and better than k means algorithm. Therefore, through these observations in this preliminary experiment, it is decided that NTSO and Kohonen Networks are adopted for implementing DDO systems. In clustering 1,000 documents, Kohonen Networks and k means algorithm took 33 and 35 seconds,

respectively. Single pass algorithm and NTSO took only less than one second and 9 seconds for clustering the number of documents.

4.4.2 Experiments in Text Categorization

This section discusses a preliminary experiment where approaches to text categorization are evaluated on three test beds. In this preliminary experiment, five approaches, SVM, NB, KNN, Back Propagation, and NTC participate, and three collections of news articles, Newspage.com, 20NewsGroups, and Reuter 21578, are used as the test beds for evaluating these approaches. In two of three test beds, the five approaches are evaluated both with decompositions of text categorization into binary classification problems as many as predefined categories and without such decompositions. This section will observe classification performance of the four approaches to text categorization.

In this preliminary experiment, documents are represented into string vectors for using NTC or numerical vectors for using the other methods. Dimensions of numerical vectors and string vectors representing documents are set as 500 and 50, respectively, in the three sets of preliminary experiment. For encoding documents into numerical vectors, most frequent 500 words from a given training set for each problem are selected as their features. The values of the features of numerical vectors are binary ones indicating the absence or presence of words in a given document. For encoding documents into string vectors, the most frequent 50 words are selected from a given document as values of its corresponding string vector.

The parameters of the five approaches involved in this preliminary experiment are set by tuning them with a validation set, which is built by selecting 600 documents randomly from training documents, spanning the three test beds. Table 10 shows the definition of the parameters which is obtained through this tuning. With the parameters defined in table 10, the five approaches to text categorization will be applied to all of three test beds. As shown in table 10, a modified version of NB, where posterior probabilities are smoothed, is used in this preliminary experiment, rather than the original version of NB. The parameter of NB, m, is for the modified version as shown in equation (21).

Table 10. Parameters of the Five Approaches

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

4.4.2.1 NewsPage.com

The first set of this preliminary experiment pursues the evaluation of the five approaches on the test bed, Newspage.com, with and without the decomposition. This test bed

consists of 1,200 news articles in the format of plain texts built by copying and pasting news articles in the web site, www.newspage.com. Table 11 shows the predefined categories, the number of documents of each category, and the partition of the test bed into training set and test set. As shown in table 11, the ratio of training set to test set is set as 7:3. Here, this test bed is called Newspage.com, based on the web site, given as its source.

Table 11. Training Set and Test Set of Newspage.com

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

The task of text categorization on this test bed is decomposed into five binary classification problems, category by category. In each binary classification problem, a classifier answers whether an unseen document belongs to its corresponding category, or not. Table 12 shows the definition of training sets of the predefined categories. In table 12, 'positive' indicates that documents belong to the corresponding category and such documents will called positive documents, while 'negative' indicates that documents do not and such documents will be called negative documents. For each training set, all of documents not belonging to its corresponding category are allocated as negative documents. For each test set, negative documents are allocated as many as positive documents defined in the third column of table 11.

Table 12. The Allocation of Positive and Negative Class in Training Set of each Category

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

Figure 28 presents the result of evaluating the five approaches on the test bed, Newspage.com, with a graph. On x-axis of the graph, the left group indicates the micro-averaged F1, the right group indicates the macro-averaged F1, and each bar within each group indicates one of the five approaches. The y-axis of the graph indicates the F1-measure which weight recall and precision, equally. The result of this evaluation shows that back propagation works best among the approaches with decomposition of the task of text categorization on this test bed into five binary classification problems. Although NTC is the second best approach to back propagation, it is comparable and competitive to back propagation, as shown in figure 28.

Figure 29 shows the result of evaluating the four classifiers except SVM without decomposition on this test bed. The reason of excluding SVM in this evaluation is that

SVM is applicable only to a binary classification problem. Without decomposition, a classifier answers one of the five categories presented in table 11 and 12, instead of yes or no. Y-axis of figure 29 indicates accuracy which is the portion of correctly classified test documents to all of them, instead of F1-measure. This result shows that NTC is the best text classifier among the four approaches on this test bed without decomposition.

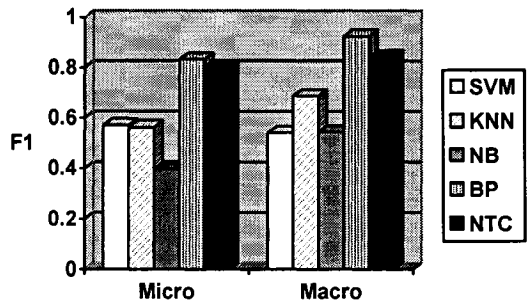

Figure 28. Result of evaluating five Text Classifiers in Newspage.com with decomposition

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

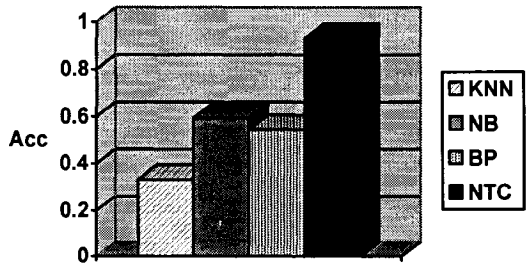

Figure 29. Result of evaluating four Text Classifiers in Newspage.com without decomposition

4.4.2.2 20NewsGroups

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

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

Figure 30 shows the result of evaluating the five approaches on the test bed, 20NewsGroup. Since each category contain identical number of test documents, micro-averaged and macro-averaged F1 are same as each other. Therefore, their performances are presented in one integrated group, instead of two separated groups, in figure 30. This result shows that back propagation is also the best approach while NB is the worst approach with the decomposition of the task on this test bed. Like the previous experiment set, NTC is comparable and competitive with back propagation.

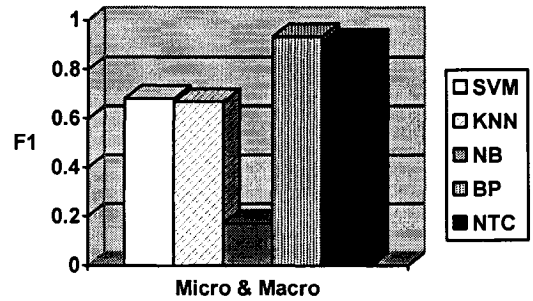

Figure 30. Result of evaluate the five text classifiers in 20Newsgroup with decomposition

Figure 31 shows the result of evaluating the four classifiers except SVM without the decomposition on this test bed. In this case, a classifier answers to each test document by providing one of 20 categories. This result shows that there exits two groups: better group and worse group. The former contains back propagation and NTC, and the latter contains NB and KNN.

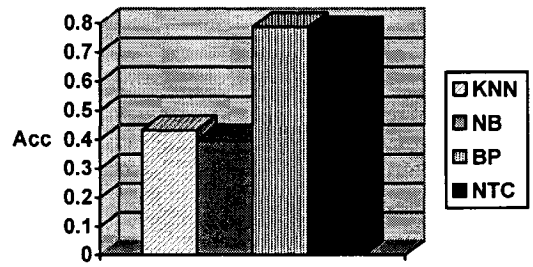

Figure 31. Result of evaluating four Text Classifiers in 20NewsGroups without decomposition

Like the previous set of this preliminary experiment, NTC is competitive with back propagation with smaller size of input data and lower number of training iterations. The result of this set is similar as that of the previous set, with respect to the trend. What we note in the results is that there is big difference between back propagation and Naïve Bayes. When representing documents into numerical vectors, a lot of noise gets introduced. Therefore, in 1995, Wiener proposed back propagation for text categorization [Winer 1995]. The advantage of back propagation over NB is that back propagation is very tolerant to noise in the training examples. However, in NB, posterior probabilities given as classifiers are very sensitive to noise. Therefore, there is big difference between the two approaches. Note that NB is more practical for learning from training examples with little noise, since the learning of back propagation is very slow.

4.4.2.3 Reuter21578

The last set of this preliminary experiment is for the evaluation of the five classifiers on the test bed, Reuter21578, which is a typical standard test bed in the field of text categorization. In this experiment set, ten largest categories are selected. Table 13 shows ten selected categories and the number of training documents and test documents in each

category. The partition of this test bed into training set and test set follows the version, ModApte, which is the standard partition of Reuter 21578 for evaluating text classifiers [Sebastiani 2002]. The number of documents in each category is very variable as shown in table 13. In this preliminary experiment, we did not evaluate these approaches without decomposition, where a classifier generate one of predefined categories, instead of answering 'belongs' or 'not belongs', since each document may have more than one category. Although it is possible to evaluate them without decomposition in this situation, it is neither easy nor simple to do that. Therefore, evaluation of these approaches without the decomposition was omitted in this set.

Table 13. Partition of Training Set and Test Set in 20NewsGroup

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

The task of text categorization on this test bed is decomposed into ten binary classification ones. For training set of each category, 2000 negative documents are allocated identically with regardless of the number of positive documents by selecting them at random from the remaining. Eight of ten categories except 'acq' and 'earn' are under imbalanced distribution over positive and negative class. For test set of each category, negative documents are allocated as many as positive documents, with its balance.

Figure 32 shows the result of evaluating the five approaches on this test bed with the decomposition. Unlike the two previous experiment sets, this result shows that NTC is the best approach among the five ones with respect to micro-averaged and macro-averaged F1. NTC has more difference from the others with respect to macro-averaged F1, as shown in the right side of figure 32.

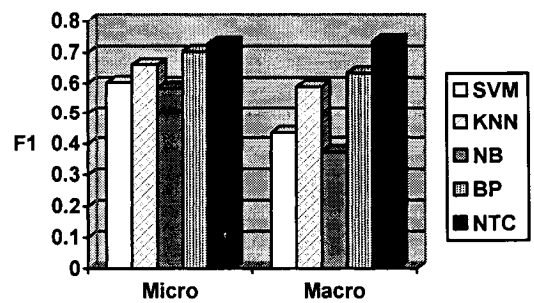

Figure 32. Result of evaluating five Text Classifiers in Reuter 21578 with decomposition

Figure 32 shows not only performance of the five approaches with respect to two evaluation measures, but also how much sensitive to sparse category containing a small number of positive training documents, these approaches are. More than half of ten categories correspond to sparse categories, as shown in table 13. Especially, SVM and NB show their large difference between their micro-averaged and macro-averaged F1 when the right side is compared with the left side of figure 32. This means that they are very sensitive to sparse categories. However, NTC shows little difference between two evaluation measures; this means that NTC is tolerant to sparse categories.

This preliminary experiment implies that NTC is most practical among the five approaches. First, NTC has an acceptable performance in spite of its far smaller size of input data and iterations of training than those of the others. Although back propagation is the best approach in two of the three test beds, NTC is competitive and comparable to back propagation in its performance and takes far less time for training than back propagation.

Second, NTC is very tolerant to sparse categories, as mentioned above. In the practical world, such sparse categories may appear very often. At this point, NTC is more practical than the others.

Third, NTC is transparent in its classification of each unseen document into a particular category, as mentioned in subsection 4.4.4.1. The reason is that NTC uses string vectors as its input vectors, which display the content of full text roughly for users. Entries of tables owned by the learning nodes are represented into symbolic classification rules. We can extract the reason for classifying an unseen document into a particular label by obtaining the corresponding rules. Since users want not only a final category of unseen document, but also the reason of such classification. NTC is very practical for doing that.

Chapter 5 Optimizing DDO: Resampling

The previous chapter proposed a new representation of documents and two neural networks using the representation for improving the performance of DDO systems. It also validated the two neural networks through a set of experiments. In DDO systems of the four-phase scenario, the class imbalance problem usually occurs, just before classifier training; documents organized just after creation mode are distributed with an imbalance over clusters. DDO systems may need resampling methods, in order to improve their performance by making the distribution of documents over clusters almost balanced. This chapter discusses resampling methods for DDO systems in four sections. The first section discusses specifically the role of resampling and the process of applying resampling methods to DDO systems. The second and third sections describe oversampling methods and an undersampling method applicable to both numerical vectors and string vectors, respectively. The last section presents the results of evaluating the resampling methods in text categorization.

5.1 Resampling for DDO Systems

This section describes the role of resampling for DDO implementations. As previously discussed, the first two phases, text clustering and cluster identification, generate a series of clusters of various sizes as an organization of documents. This means that the distribution over clusters in the organization is usually imbalanced. The largest cluster is called the *majority cluster* and the smallest cluster is called the *minority cluster*. In DDO implementations, resampling is defined as the process of reducing the size of the majority cluster – *undersampling* – or increasing that of the minority cluster – *oversampling* toward a balanced distribution of all the clusters. Since in the three-phases-scenario, the arrangement of added documents is based on their similarity with the prototype of each cluster, the size of the cluster does not influence such an arrangement. Therefore, the three-phases-scenario of DDO implementations does not need resampling. In this section, the role of resampling for DDO implementations is restricted to the four-phases-scenario.

A disadvantage of the four-phases-scenario is that a classifier trained using an unbalanced set of sample documents may build a strong bias toward the majority cluster in the classifier training phase. In the document classification phase added documents may be arranged into the majority cluster with highest probability. Until the reorganization of documents occurs, the difference between the majority and the minority cluster increases gradually. The arrangement of added documents may end up being determined according to the size of the clusters rather than by their content. The four-phases-problem needs to use resampling in order to solve its disadvantage.

Figure 33 illustrate the process of resampling in DDO systems implemented based on the four phases scenario. Resampling is performed just after creating an organization of documents through text clustering and cluster identification. The organization of documents consisting of clusters of various sizes is given as the input. At first, the average size of the clusters is obtained. For each cluster, if its size is larger than the average size, its size is reduced to the average size by undersampling, which is the process of removing sample documents in a particular cluster at random. Otherwise, its size is increased to the average size by oversampling, which is the process of adding more encoded sample documents to the given cluster. A series of clusters of the same size is generated from this process. This resampling for the four-phases-scenario creates

clusters of the average size. Its goal is to build a balanced distribution of sample documents over clusters without changing the total number of sample documents. An increased number of sample documents which is caused by oversampling, reduces the speed of the classifier training phase. A reduced number of sample documents which is caused by undersampling, leads to an information loss of sample documents. In order to avoid the two problems, the number of sample documents is maintained by using both oversampling and undersampling.

Resampling
Input: A Set of Clusters of various sizes
Step 1: Compute the average size of clusters
Step 2: For each cluster
Step 2-1: If the size of the cluster is larger than the average size, reduce its size to the average size by undersampling
Step 2-2: otherwise, increase its size to the average size by oversampling
Output: A Series of documents clusters of the same size

Figure 33. The Process of Resampling for the Four-Phases-Scenario

5.2 Oversampling Methods

This section describes four oversampling methods: simple oversampling, crossover based oversampling, mutation based oversampling, and SMOTE [Chawla et al 2000]. Cross over based oversampling and mutation based oversampling are the simplified versions of SMOTE, using the concept of the recombinant operators of simple genetic algorithm: crossover and mutation [Goldberg 1989]. Note that all the oversampling methods except simple oversampling are different from the original version of SMOTE, because they are modified to be applicable to documents encoded as string vectors. In the DDO implementation, oversampling is defined as the process of generating more encoded sample documents based on existing ones. Additional encoded sample documents generated through this process are called virtual encoded documents, since they have no actual document that they encode. The original encoded sample documents are called actual encoded documents, since they have their actual documents that they encode. The role of oversampling in the four-phases-scenario for DDO implementation is to increase the size of minority clusters toward a balanced distribution of clusters in the organization.

5.2.1 Simple Oversampling

The first method is simple oversampling, where more training examples are generated by copying some of existing ones. This method is the simplest among the four oversampling methods described in this subsection. All the virtual encoded documents generated, using this method, are identical to some existing ones. This method is applicable to any type of encoded documents; it is applicable to any one of the three strategies of encoding documents described in sections 2.1 and 4.1.

Figure 34 illustrates the process of increasing the size of a minority cluster to the average size of clusters in the four-phases-scenario, using simple oversampling. The minority cluster and the average size of clusters are given as its input. The minority cluster corresponds to the cluster whose size is less than the average size. The number of virtual encoded documents to generate is determined by the difference between the average size of clusters and the particular cluster's size. Each virtual encoded document

is produced by selecting one of the actual encoded documents at random, copying it and adding it to the cluster. The output of this process is the cluster whose size is increased to the average size of clusters: the cluster including both actual encoded documents and virtual encoded documents. Virtual encoded documents are the same as some of actual encoded documents in this oversampling method.

Simple Oversampling
Input: A minority cluster consisting of actual encoded documents and the average size of the clusters
Step 1: Determine the number of virtual encoded documents to produce
#virtual encoded documents = the average size of the clusters – the size of the cluster
Step 2: For 1 to #virtual encoded document
Step 2-1: Select one of the encoded documents given as an input
Step 2-2: Generate its duplicated document by copying it
Step 2-3: Add it to the list of encoded documents
Output: The cluster consisting of actual encoded documents and virtual encoded documents

Figure 34. The Process of Simple Oversampling for Minority Classes

5.2.2 Crossover based Oversampling

The second oversampling method is crossover-based oversampling, where two training examples are selected at random and an arbitrary example between them is created as an additional example. The reason for calling this method crossover-based oversampling is that oversampling is achieved by obtaining an arbitrary hybrid between two existing examples. This method is also simple and it changes not only the number of training examples but also their distribution. But it is applicable to only two of our three types of document encoding: numerical vectors and string vectors; it is not applicable to documents encoded as bags of words.

Crossover-based Oversampling
Input: A minority cluster consisting of actual encoded documents and the average size of clusters
Step 1: Determine the number of virtual encoded document to produce
#virtual encoded document = the average size of clusters – the size of the cluster
Step 2: For 1 to #virtual encoded document
Step 2-1: **Select two of the encoded documents given as an input at random**
Step 2-2: **Generate an encoded document between them randomly**
Step 2-3: Add it to the list of encoded documents
Output: The cluster consisting of actual encoded documents and virtual encoded documents

Figure 35. The Process of Cross-over based Oversampling for a Minority Cluster

Figure 35 illustrates the process of crossover-based oversampling to increase the size of a minority cluster. The input and the output of this process are the same as those of simple oversampling. In figure 35, step 2-1 and step 2-2 are different from those of figure 34 that illustrates the process of simple oversampling. If the documents are encoded into numerical vectors, two encoded documents selected in step 2-1 are denoted by $d_i^f = [\phi_i(w_1) \quad \phi_i(w_2) \quad .... \quad \phi_i(w_n)]$ and $d_j^f = [\phi_j(w_1) \quad \phi_j(w_2) \quad .... \quad \phi_j(w_n)]$. If a virtual

encoded document is denoted by $d_a^f = [\phi_a(w_1) \ \phi_a(w_2) \ .... \ \phi_a(w_n)]$, each of its elements, $\phi_a(w_k)$, $1 \le k \le n$, is computed using equation (39).

$$\phi_a(w_k) = \begin{cases} \phi_i(w_k) + \gamma(\phi_j(w_k) - \phi_i(w_k)) \text{ if } \phi_j(w_k) \ge \phi_i(w_k) \\ \phi_j(w_k) + \gamma(\phi_i(w_k) - \phi_j(w_k)) \text{ otherwise} \end{cases} \quad (39)$$

where, γ is a random normalized value between zero and one. If documents are encoded into string vectors, two documents selected in step 2-1 are denoted by $d_i^s = [w_{i1}, w_{i2}, ..., w_{in}]$ and $d_j^s = [w_{j1}, w_{j2}, ..., w_{jn}]$. In this type of document encoding, an virtual encoded document is denoted by $d_a^s = [w_{a1}, w_{a2}, ..., w_{an}]$, and each of its element, w_{ak} $1 \le k \le n$, is obtained using equation (40).

$$w_{ak} = \text{rand}(I(w_{ik}, w_{jk})) \quad (40)$$

Before using crossover based oversampling on string vectors, the similarity matrix should be defined from a corpus. Each element of a string indicating an encoded virtual document is an arbitrary element of the set of inter-words between elements of two selected string vectors indicating actual documents.

5.2.3 Mutation based Oversampling

The third method is mutation based oversampling, where an additional example is obtained by mutating each element of a particular existing example. This method is also simpler than crossover based oversampling because only one example is selected, and the method changes both the number of examples and their distribution, like crossover-based oversampling. The reason for calling this method mutation based oversampling is that each element of an existing example is modified to generate an additional example. This method is also applicable to numerical vectors and string vectors, like crossover based oversampling.

Mutation based Oversampling
Input: A minority cluster consisting of actual documents and the average size of clusters
Step 1: Determine the number of virtual document to produce
#virtual document = the average size of clusters – the size of the cluster
Step 2: For 1 to #virtual document
Step 2-1: **Select an encoded documents given as an input at random**
Step 2-2: **Generate an encoded document by modifying each of its element**
Step 2-3: Add it to the list of encoded documents
Output: The cluster consisting of actual documents and virtual documents

Figure 36. The Process of Mutation based Oversampling for a Minority Cluster

Figure 36 illustrates the process of increasing the size of minority cluster using mutation based oversampling. In this method, only one existing actual document is selected to generate a virtual document. When encoding documents into numerical vectors, the actual document selected through step 2-1 is denoted by $d_i^f = [\phi_i(w_1) \ \phi_i(w_2) \ .... \ \phi_i(w_n)]$. Each of the elements of the encoded virtual document denoted by $d_a^f = [\phi_a(w_1) \ \phi_a(w_2) \ .... \ \phi_a(w_n)]$, $\phi_a(w_k)$, $1 \le k \le n$, is computed using equation (41),

$$\phi_a(w_k) = \phi_i(w_k) + \gamma \quad 0 \le \gamma \le \gamma_{\max} \text{ (41)}$$

where γ is a random value between zero and $\gamma_{\max}$, which is given as a parameter for this process only in this encoding. When encoding documents into string vectors, an encoded actual document selected through step 2-1 is denoted by $d_i^s = [w_{i1}, w_{i2}, ..., w_{in}]$. Each of the elements of the encoded virtual document denoted by $d_a^s = [w_{a1}, w_{a2}, ..., w_{an}]$, w_{ak} $1 \le k \le n$, is obtained using equation (42),

$$w_{ak} = sim(w_{ik}) \text{ (42)}$$

where $sim(\cdot)$ is a function that takes a word and finds the most similar word from a similarity matrix. The most similar word corresponds to the word which has the highest similarity with the original word in a similarity matrix. Before using this method on string vectors, a similarity matrix should be built from a corpus. In this method on string vectors, a virtual document is obtained by replacing the words as given elements of the actual document by their most similar words.

5.2.4 SMOTE

The last method of oversampling is SMOTE (Synthetic Minority Oversampling Technique), where crossover based oversampling is applied to two examples selected not from the category, but from several nearest examples [Chawla, et al. 2000]. This method was only used on numerical vectors in a particular binary classification problem not in text categorization. In this oversampling method, two examples are selected from examples which were selected previously as nearest ones to a random example. Although this method is applied only to numerical vectors in the literature, [Chawla, et al. 2000], it is also applicable to string vectors.

SMOTE
Input: A minority cluster consisting of actual encoded documents, the average size of clusters, and **k**
Step 1: Determine the number of virtual encoded virtual document to produce
#encoded virtual document = the average size of clusters – the size of the cluster
Step 2: For 1 to #encoded virtual document
Step 2-1: Select an encoded document at random
Step 2-2: Select k most similar encoded documents to the document selected in step 2-1
Step 2-3: Select two of the encoded documents at random from k most similar sample documents selected in step 2-2
Step 2-4: Derive another numerical vector or string vector from the selected documents by equation (39) or (40)
Step 2-5: Add it to the list of encoded documents
Output: The cluster consisting of actual encoded documents and virtual encoded documents

Figure 37. The Process of SMOTE for a Minority Cluster

Figure 37 illustrates the process of increasing a minority cluster using SMOTE. In this method, k is given additionally as its parameter. In step 2-1, an encoded document is selected at random from the sample documents. In step 2-2 of figure 37, k most similar actual documents are selected, where k is given as a parameter as discussed previously. The process of computing their similarities in three types of document encodings was already described in section 2.1. In step 2-3, among k most similar documents, two encoded sample document are selected at random. Crossover based oversampling is

applied to these two encoded sample documents in order to generate an encoded virtual document. Step 2-3 and step 2-4 were already described in subsection 3.2.2.

SMOTE generates virtual documents more similar to the actual documents than cross over based oversampling. This is because SMOTE selects k most similar actual documents to a random actual document in advance as the scope of selecting two actual documents in order to induce a virtual document. In other words, crossover based oversampling has the broader scope of selecting a pair of actual documents for generating a virtual document than SMOTE. In crossover based oversampling, two far different examples within a cluster may be selected accidentally and induce a virtual document different from the actual documents. Such a virtual document is not useful for the classifier training phase, because it may build a small disjunct. Therefore, SMOTE provides more useful virtual documents than crossover based oversampling.

5.3 Undersampling Method

In the DDO implementation, *undersampling* is defined as the process of selecting some existing sample documents and removing them. The role of undersampling in the four-phase-scenario for DDO implementation is to reduce the size of the majority clusters toward a balanced distribution of clusters in the organization.

5.3.1 Simple Undersampling

Simple undersampling is the method in which some of the given training examples are selected and removed at random. This method is the simplest of the three undersampling method and is applicable to any type of document encoding. Important training examples however may be lost and this may contribute a degradation of the performance of classifiers.

Figure 38 illustrates the process of removing some sample documents from a majority cluster using simple undersampling. A majority cluster and the average size over clusters are given as input. The number of sample documents to be removed is determined by the difference between its size and the average size. The process of selecting and removing an encoded sample document randomly is repeated until this number is reached. The output of this process is the cluster whose size got reduced to the average size.

Simple Undersampling
Input: A majority cluster and the average size of clusters,
Step 1: Determine the number of sample documents that needs to be removed
#documents for their removal = the size of cluster - the average size of clusters
Step 2: For 1 to # documents for their removal
Step 2-1: Select an sample document at random
Step 2-2: Remove it from the cluster
Output: The downsized cluster

Figure 38. The Process of Simple Undersampling for a Majority Cluster

5.4 Experiment with Resampling

This section shows experimental results of evaluating resampling methods applied to text classifiers. Like the experiments in section 4.4.2, the three test beds, NewsPage.com, 20NewsGroups, and Reuter21578, and the five approaches to text categorization, NB,

SVM, KNN, back propagation, and NTC are used for this preliminary experiment. The parameters involved in the five approaches are set as defined in table 10. The partitions of the three test beds into training sets and test sets are identical to the preliminary experiment in section 4.4.2. The five resampling methods described in section 5.2 and 5.3, simple oversampling, crossover oversampling, mutation oversampling, SMOTE, and simple undersampling participate in their evaluations.

Figure 39 presents the results of evaluating the five resampling methods applied to the five approaches to text categorization on the first test bed, NewsPage.com. In figure 39, the left horizontal bar graph indicates the results of evaluating the five resampling methods with micro-averaged F1, and the right one indicates those of evaluating them with macro-averaged F1. In each horizontal bar graph, the x-axis means the value of micro-averaged or macro-averaged F1. In the y-axis, each group indicates one of the five approaches to text categorization and an individual horizontal bar within each group indicates one of the five resampling methods or non-resampling. From figure 39 to figure 41, six labels of horizontal bars in each of these figures, 'None', 'Over', 'Cross', 'Mutate', 'SMOTE', and 'Under', means 'not resampling', 'simple oversampling', 'crossover oversampling', 'mutation oversampling', 'SMOTE', and 'simple undersampling', respectively.

The results presented in figure 39 show that the five resampling methods have much effect on KNN and NB, but little effect on SVM, back propagation, and NTC. Among NB and KNN, these resampling methods have a significant effect on NB and KNN. In other words, KNN received the benefit for its performance from all of the five resampling methods, while NB received the benefit from some of them. Especially SMOTE is harmful to NB on this test bed, as illustrated in figure 39.

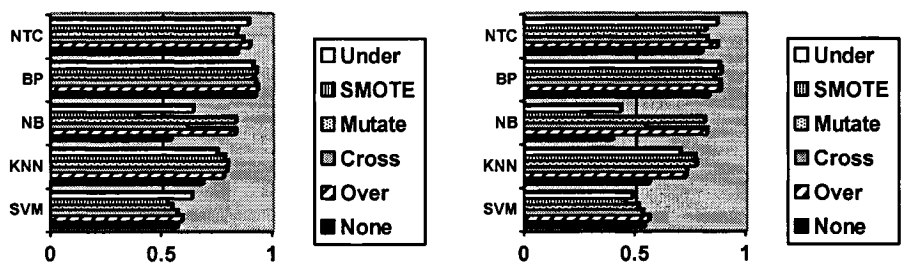

Figure 39. Results of evaluating Resampling Methods on NewsPage.com (Left: Micro averaged F1 and Right: Macro averaged F1)

Figure 40 presents the results of evaluating the five resampling methods on the second test bed, 20NewsGroups. Since all of the twenty categories of this test bed have identical numbers of documents in its test set, micro-averaged F1 and macro-averaged F1 have their same value. Therefore, these results are presented in a horizontal bar graph, instead of two horizontal bar graphs. Like the results on the previous test bed, the five resampling methods are effective much on NB and KNN, but little on the others. A difference from the previous results is that the five resampling methods are most useful for NB, among the five approaches, as shown in figure 40.

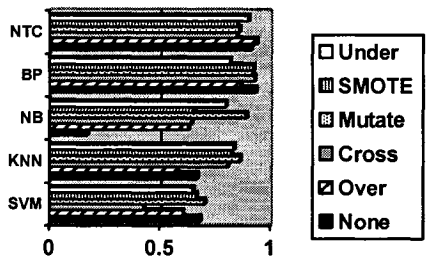

Figure 40. Results of evaluating Resampling Methods on 20NewsGroups(Micro & Macro averaged F1)

Figure 41 shows the results of evaluating the five resampling methods on the third test bed, Reuter21578, selecting most frequent ten categories. Since three of ten categories, 'acq', 'earn', and 'money fx' have their sufficient numbers of positive documents as shown in table 13, the three categories of Reuter 21578 are excluded for applying the five resampling methods; they are applied to seven of ten categories. Since macro-averaged F1 reflects the change of performance by applying the five resampling methods, more clearly, than micro-averaged F1, we present the results of evaluating them only with macro-averaged F1. The x-axis in figure 41 indicates a value of macro-averaged F1. Like the results on the two previous test beds, the five resampling methods are effective on NB and KNN. In this test bed, they are also effective slightly on SVM and back propagation. To NTC, simple oversampling and simple undersampling are slightly useful, but the others are very harmful, as shown in figure 41.

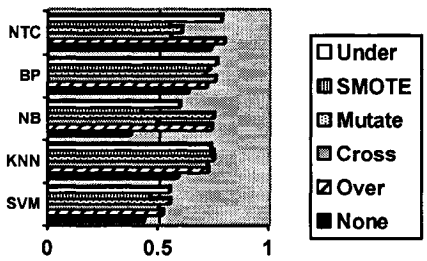

Figure 41. Results of evaluating Resampling Methods on Reuter21578 (Macro averaged F1)

What is most significant in this preliminary experiment is that the three oversampling methods, cross-over oversampling, mutation oversampling, and SMOTE, were not useful for NTC. As described in section 5.2, a similarity matrix from a corpus should be built in order to apply the three oversampling methods for string vectors. In this preliminary experiment, a similarity matrix is built from a training set of each test bed; three different similarity matrices are used for the preliminary experiment on the three test beds. In building a similarity matrix, the more documents including two words, the higher their semantic similarity regardless of target labels of such documents; documents including both words may have their different target categories each other. Especially in the test bed, Reuter 21578, more documents span over all of the predefined categories than any

other test bed, since each document has more than one category. These oversampling methods generate artificial string vectors including words informative for other categories than its target category. Since the artificial string vectors lead to discrimination loss for classification of NTC, the three oversampling methods degraded the classification performance of NTC.

Based on the results of the experiments in section 4.4.2 and 5.4, two of the five text classifiers, NTC and KNN, will be selected for implementing DDO systems. Without any resampling method, NTC shows better performance than SVM, NB, and KNN, and its competitive performance to back propagation, in spite of its smaller input size and smaller number of iterations. Since resampling methods are little useful or harmful to NTC as shown in this preliminary experiment, we will not apply any resampling method in using NTC for implementing DDO system. The reason of selecting KNN as an approach to implementing DDO systems is that its performance was improved sufficiently and stably by any of the five resampling methods. Since mutation oversampling is most effective on KNN among the five resampling methods, it will be selected for its integration with KNN in implementing DDO systems.

Chapter 6 Experimental Results of DDO System

This chapter concerns the three main experiments which provide directions for developing DDO systems, as a real system. In section 6.1, the approaches to text clustering and text categorization will be selected for implementing DDO systems based on the results of the experiments in chapter 4 and 5. In section 6.2, the two scenarios of DDO systems will be compared with each other and it will be determined which of scenario is more desirable for developing DDO systems. In section 6.3, the two paradigms of document organization, SDO and DDO, will be compared with each other and DDO will be confirmed as the more desirable paradigm of document organization, than SDO. In section 4.4, we will determine whether resampling methods improves the performance of DDO systems or not, by comparing two versions of DDO system; versions without and with a resampling method.

6.1 Versions of DDO System

DDO systems may be implemented in their various versions, depending on which of unsupervised and supervised learning algorithms are used as approaches to text clustering and text categorization and whether a resampling method is used or not. The performance of a DDO system is dependent upon which versions it is implemented with. Before implementing DDO systems, we should define their versions by determining which machine learning algorithms are used as approaches to text clustering and text categorization as well as which resampling methods to use if any. In this section, versions of DDO system will be defined based on the results of the three preliminary experiments in the previous chapter, before performing the three main experiments on the evaluation of DDO systems.

The four factors that should be considered for selecting machine learning algorithms as approaches to text clustering and text categorization given as components of DDO system as the following.

- **Clustering Speed**

This factor indicates how fast documents are organized or reorganized in creation mode of DDO system.

- **Clustering Performance**

This factor indicates how well a DDO system organizes documents in its creation mode. It influences on the quality of managing documents in a DDO system.

- **Learning Speed**

This factor is applicable to DDO systems of four-phase scenario. This means how fast a classifier is trained with organized documents.

- **Classification Performance**

This factor indicates how well a DDO system arranges added documents in its maintenance mode. Together with clustering performance, it influences the quality of managing documents.

Clustering speed and learning speed are important factors for implementing real time DDO systems. Considering the four factors, we select Kohonen Networks and NTSO as the approaches to text clustering, and KNN and NTC as the approaches to text categorization, through the three preliminary experiments in chapter 4. As shown in experiment of chapter 4, NTSO is practical with respect to both clustering speed and

clustering performance, and Kohonen Networks is best clustering performance although its clustering speed is low. NTC is most practical with respect to learning speed and classification performance. KNN received benefit from resampling methods; the purpose of adopting this approach is to observe the effect of resampling in implementing DDO systems.

In table 14, the five versions of DDO system involved in the three main experiments in this chapter are defined: two versions for three-phase scenario and three versions for four-phase scenario. The second column of table 14 enumerates notations of the five versions of DDO system. The prefix, 'DDO', of all notations, means DDO system. The character, '3' or '4', following 'DDO' means three-phase scenario or four-phase scenario, respectively. In the five versions of DDO system, 'K' and 'N' just after the character, '3' or '4', mean Kohonen Networks and NTSO, respectively. In the two versions of four-phase scenario, 'DDO4KK' and 'DDO4NN', 'K' and 'N' given as their last characters mean K nearest neighbor and NTC. In the version of four-phase scenario, 'DDO4KKO', the second last character, 'K' and the last character, 'O', mean K nearest neighbor and oversampling, respectively. For example, the notation, 'DDO4NN', means a version of DDO system of four-phase scenario where NTSO and NTC are used as approaches to text clustering and text categorization, respectively.

When we adopt NTSO as the approach to the component, text clustering, we must build a similarity matrix from a given collection of documents. In early stage, whenever the status of the DDO system moves back to its creation mode, we must build a similarity matrix. However, since the number of documents is small in the early stage, it does not take much time. When a collection of documents becomes large, the change of documents from deletions and additions of documents gets weak. We can reuse the similarity matrix continually without rebuilding a similarity matrix.

Table 14. Versions of DDO Systems

Three Phases Scenario	DDO3K	DDO	Dynamic Document Organization
		3	Three Phases Scenario
		K	Kohonen Network
	DDO3N	N	NTSO
Four Phases Scenario	DDO4KK	4	Four Phases Scenario
		K	Kohonen Network
		K	K Nearest Neighbor
	DDO4KKO	O	DDO4KK + Oversampling
	DDO4NN	N	NTSO
		N	NTC

6.2 Evaluation Measure: Clustering Index

So far, we evaluated each component of our DDO system separately – we will soon discuss the evaluation of the overall DDO system. This section describes a new evaluation measure for DDO systems, which will be adopted for comparing several versions of DDO systems. The versions will include different approaches to text

clustering and text categorization as well as different strategies for encoding documents. Since the organization of documents in such systems is always variable, the evaluation measures of text categorization such as accuracy, recall, precision, and F1 measure are not suitable for the their evaluation. Such evaluation measures are applicable only to the situation, where the organization of documents is fixed.

In previous research on text clustering, the result of text clustering was evaluated with only labeled documents using the evaluation measures of text categorization. Supervised learning algorithms have been used to evaluate the result of text clustering [Martin 1995]. Since supervised learning algorithms, themselves, are not completely accurate classifiers, the evaluation using them can not be expected to be accurate. In 1997, J. Aslam and his colleagues used recall, precision, and F1 measure as the evaluation measures for text clustering [Aslam, et al. 1997]. As mentioned above, these measures are applicable only if the number of clusters is same to that of the target categories. In 1998, M. Goldszidt and M. Sahami labeled clusters based on the majority of each cluster, and evaluate the clustering result using the consistency between the clustering result and the desired distribution [Goldszidt and Sahami 1998]. This measure is applicable only to the corpus of labeled documents and it does not consider the situation where a target category is partitioned into two clusters in the result of text clustering. In 2001, T. Jo proposed an evaluation method that is applicable to a corpus of unlabeled documents [Jo 1998]. This evaluation measure is based on the inter-cluster entropy and the intra-cluster entropy computed using word frequencies. In WEBSOM, the result of text clustering is not evaluated but demonstrated visually [Kohonen, et al. 2000][Kaski, et al. 2000]. Other works on text clustering did not evaluate the result of text clustering but also showed its visualized results [Mostafa and Lam 2000][Hatzivassiloglou, et al. 2000].

Recently in 2006, Tan and his colleagues mentioned evaluation of clustering entities in detail in their textbook [Tan et al 2006]. Their literature provides basic idea for deriving an integrated evaluation measure of text clustering, considering two factors, cohesion and isolation, at same time. *Cohesion* indicates how homogeneous entities are within each cluster, while *isolation* indicates how heterogeneous entries are between clusters. For each factor, an equation was defined based on proximities of entities. Proximity of entities was defined roughly as their similarities or their distances. However an integrated evaluation measure of text clustering was not derived, proximities of entities were not defined specifically, and the equations of computing values of cohesion and isolation did not provide normalized values. Therefore, this research will address the limit of their proposal concerning evaluation of clustering by integrating the two factors into an integrated measure providing a normalized value and defining proximities of documents specifically.

The first subsection proposes an evaluation measure, called clustering index, for text clustering in the situation where the number of clusters is different from that of the target categories. This measure is relevant when a corpus of labeled documents is used as a test bed.

6.2.1 Clustering Index

This section describes an evaluation measure of text clustering when using exclusively labeled documents as test data. The evaluation measure is based on the ideas in two literatures, [Tan et al 2006] and [Duda et al 2001]. Cohesion means the purity of

homogenous entities within each cluster, and corresponds to *intra-cluster similarity* mentioned in [Duda et al 2001], while isolation means the discrimination among clusters, and corresponds to *inter-cluster similarity* mentioned in [Duda et al 2001]. We will use in this dissertation 'intra-cluster similarity' and 'inter-cluster similarity' for deriving an evaluation measure of text clustering, instead of 'cohesion' and 'isolation'. Therefore, the ultimate objective of clustering is to maximize intra-cluster similarity and minimize inter-cluster similarity. This section will derive an evaluation measure based on these two kinds of similarities, towards this objective. Here, proximities of documents are defined as consistency between two documents in their target label; if two documents have identical label, their proximity is 1.0, and otherwise, it is 0.0.

Using a corpus of labeled documents for the evaluation of text clustering, the similarity between two documents is a binary value, zero or one. If two documents belong to the same target category, c_t, the similarity between them is 1.0. Otherwise, the similarity is 0.0. The process of computing the similarity between two labeled documents, d_i and d_j is expressed with equation (43).

$$sim(d_i,d_j)=\begin{cases}1 & \text{if } d_i,d_j \in c_t \\ 0 & \text{otherwise}\end{cases} \quad (43)$$

Let's assume that there are $|c|$ clusters in the given result. A cluster c_k $1 \le k \le |c|$ includes a series of documents and is denoted as a set of documents by $c_k=\{d_{k1},d_{k2},...,d_{k|c_k|}\}$. The intra-cluster similarity of the cluster, c_k, σ_k is computed using equation (44) and indicates the average similarity of all pairs of different documents included in the cluster, c_k.

$$\sigma_k=\frac{2}{|c_k|(|c_k|-1)}\sum_{i>j} sim(d_{ki},d_{kj}) \quad 1 \le i,j \le |c_k| \quad (44)$$

If a set of clusters as the result of text clustering is denoted by $C=\{c_1,c_2,...,c_{|C|}\}$, the average intra-cluster similarity, $\bar{\sigma}$ is computed using equation (45), by averaging the intra-cluster similarities of the given clusters.

$$\bar{\sigma}=\frac{1}{|C|}\sum_{k=1}^{|C|}\sigma_k \quad (45)$$

The inter-cluster similarity between two clusters, c_k and c_l, δ_{kl}, is computed using equation (46) and indicates the average similarity of all possible pairs of two documents belonging to their different clusters.

$$\delta_{kl}=\frac{1}{|c_k||c_l|}\sum_{i=1}^{|c_k|}\sum_{j=1}^{|c_l|} sim(d_{ki},d_{lj}) \quad (46)$$

The average inter-cluster similarity $\bar{\delta}$ is computed using equation (47), by averaging all possible pairs of different clusters.

$$\bar{\delta}=\frac{2}{|C|(|C|-1)}\sum_{k>l}\delta_{kl} \quad (47)$$

From equation (43) to equation (47), the average intra-cluster similarity, $\overline{\sigma}$ and the average inter-cluster similarity, $\overline{\delta}$, over the given clusters are obtained. Then, the clustering index, CI can be computed using equation (48).

$$CI = \frac{\overline{\sigma}^2}{\overline{\sigma} + \overline{\delta}} \quad (48)$$

Equation (48) shows that a normalized value between zero and one is given in the clustering index. If the average intra-cluster similarity is large and the average inter-cluster similarity is small, the value of the clustering index is close to one. On contrary, if the average intra-cluster similarity is small and the average inter-cluster similarity is large, its value is close to zero. If the average intra-cluster similarity and the average inter-cluster similarity are almost identical, its value is 0.5 indicating a random clustering. If the clustering index is less than 0.5, it is indicated that the result is worse than a random clustering.

6.3 Three vs Four-phase scenarios

In this section, two scenarios of DDO system, three-phase scenario and four-phase scenario, are compared with each other with respect to the quality of managing documents. The two versions of three-phase scenario, 'DDO3K' and 'DDO3N', and the two versions of four-phase scenario, 'DDO4KK' and 'DDO4NN', will participate in this experiment. The parameters of the involved machine learning algorithms are set as defined in table 8 and table 10. Two collections of news articles, 'NewsPage.com' and '20NewsGroups', are used as test beds for evaluating the four versions of DDO system. The goal of this experiment is to determine which of scenario should be adopted for developing DDO systems.

6.3.1 NewsPage.com

This section concerns the set of this experiment where the two scenarios are evaluated on the test bed, NewsPage.com. Three manipulations, 'regular additions', 'irregular additions', and 'addition & deletion', are performed on the four versions of DDO system in this experiment set. Each manipulation consists of five steps as illustrated in table 15. In each step, documents are added or deleted with their complete balance over five categories given in the collection. For example, in step 3 of irregular addition in table 15, the addition of 200 documents means the addition of 40 documents per category. The clustering index is used to measure the final quality of managing document after the five steps in each manipulation.

Table 15. The Manipulations of Documents on NewsPage.com

	Step 1	Step 2	Step 3	Step 4	Step 5
Regular Addition	100	100	100	100	100
Irregular Addition	100	125	200	50	25
Addition and Deletion	100	-25	125	-50	100

The condition of the transition from maintenance mode to creation mode is whether the current number of transactions is greater than the transaction threshold given as a parameter to the DDO system. The current number of transactions corresponds to the number of documents added or deleted in maintenance mode of DDO system. If a DDO system enters creation mode from maintenance mode, its current number of transactions is reset. In this experiment set, the transaction threshold is set to 25 which is the minimum number of documents among steps in table 15. Therefore, the versions of the DDO system moves back to creation mode if their current number of transactions is greater than 25.

Figure 42 presents the results of evaluating two scenarios on the test bed, NewsPage.com with the two graphs. In figure 42, the left graph corresponds to three-phase scenario, and the right graph corresponds to four-phase scenario. The y-axis of the graphs means a value of clustering index to the results of organizing and maintaining documents, after the five steps of each manipulation. On the x-axis, each group indicates a version of the DDO system and an individual graph means a manipulation of each version of DDO system. The two graphs in figure 42 show that the two versions of the four-phase scenario have much higher clustering indices in every manipulation than the two versions of the three-phase scenario.

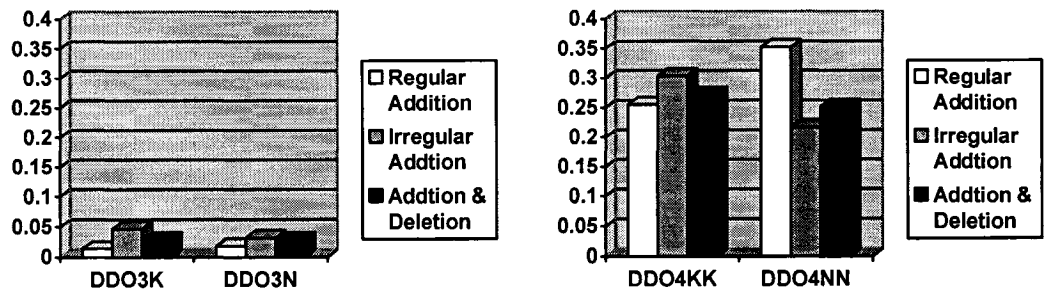

Figure 42. Clustering Index of Two Scenarios of DDO Systems in NewsPage.com

In figure 43, the dominance of the versions of four-phase scenario over those of three-phase scenario is visualized with a pie chart. In the pie chart, the white area indicates the mean clustering index corresponding to four-phase scenario which is averaged over six clustering indices of the right graph in figure 42, while the black area indicates mean clustering index corresponding to three-phase scenario which is averaged over six clustering indices of the left graph. The mean clustering index of four-phase scenario is 0.2753, while that of three-phase scenario is 0.0279. This experiment set shows finally that the two versions of the four-phase scenario manage documents ten times better than those of the three-phase scenario on the test bed, NewsPage.com.

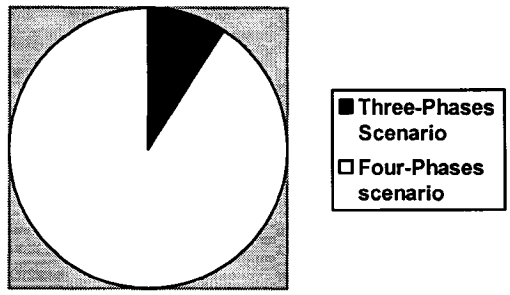

Figure 43. Three Phases Scenario vs Four Phases Scenario in NewsPage.com

6.3.2 20NewsGroups

In this section, two scenrios of DDO system will be evaluated on another test bed, '20NewsGroups'. Table 16 illustrates manipulations of documents to the four versions of DDO system on the test bed, 20NewsGroups. Among 20 categories of this test bed, we use only four representative categories like the preliminary experiment on text clustering in section 3.3.3. In every step contained in table 16, documents are added or deleted with their complete balance over the four categories. In this experiment set, the transaction threshold is set at 20 which is the minimum number of documents among steps in table 16.

Table 16. The Manipulations of Documents on 20NewsGroups

	Step 1	Step 2	Step 3	Step 4	Step 5
Regular Addition	100	100	100	100	100
Irregular Addition	100	120	200	60	20
Addition and Deletion	100	-20	120	-40	100

Figure 44 presents the results of evaluating two scenarios on the test bed, '20NewsGroups'. The results are very similar as those on the previous test bed, 'NewsPage.com', with respect to their trends. The results on this test bed show also that the two versions of the four-phase have much higher clustering indices than those of the three-phase scenario in every manipulation.

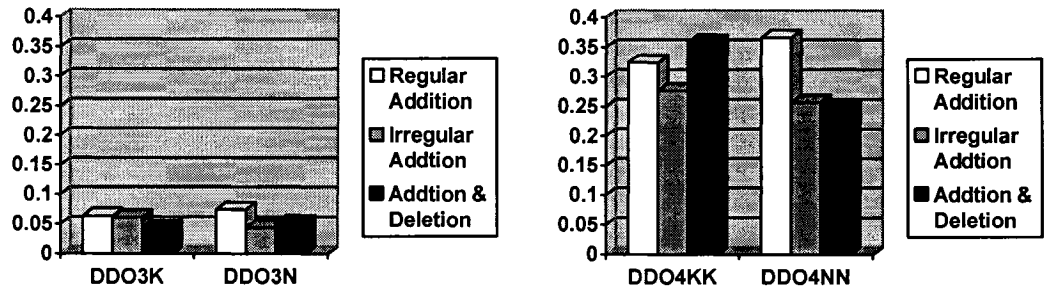

Figure 44. Clustering Index of Two Scenarios of DDO Systems in 20NewsGroups

In figure 45, the dominance of four-phase scenario over three-phase scenario is visualized with a pie chart. The mean clustering index of four-phase scenario is 0.3062, while that of three-phase scenario is 0.0565. Although the dominance of the four-phase scenario is reduced slightly in this test bed, the results still show that the versions of the four-phase scenario manage document much better than those of the three-phase scenario.

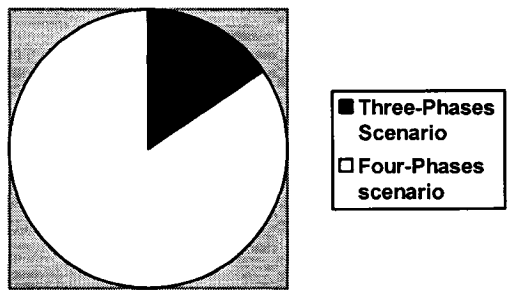

Figure 45. Three Phases Scenario vs Four Phases Scenario in 20NewsGroups

6.3.3 Discussion

The results of evaluating the two scenarios of DDO system on the two test beds show clearly that versions of four-phase scenario manage document much better than those of three-phase scenario in quality. In versions of three-phase scenario, documents are arranged based on their similarities with prototypes of clusters in maintenance mode. This means that three-phase scenario assumes implicitly that each cluster is a hyper-sphere centered on its prototype, although it is actually an irregular shape. In versions of four-phase scenario, documents are arranged by a classifier trained with organized documents. This means that each cluster is approximated more closely to its actual shape than in versions of three-phase scenario. Hence, versions of the three-phase scenario have more risk of misarranging documents in maintenance mode than those of four-phase scenario.

However, a disadvantage of versions of the four-phase scenario is that they require more time for training their classifiers than those of three-phase scenario; there exists a delay for the transition from creation mode to maintenance mode. Since the results of this experiment show that the versions of the three-phase scenario are not even competitive,

the four-phase scenario will be adopted as a direction for developing DDO systems, in spite of its disadvantage.

6.4 SDO vs DDO

This section concerns the experiment where the two paradigms of document organization, SDO and DDO, are compared with each other with respect to the quality of managing documents. Since the results of the previous main experiment show that the versions of the three-phase scenario manages documents with poor quality, compared with those of four-phase scenario, the versions of the three-phase scenario will be discarded in this experiment. In other words, SDO and DDO will be compared with each other within only the four-phase scenario. We will make manipulations identical to those illustrated in table 15 and 3 to both the SDO systems and the DDO systems in this experiment.

6.4.1 NewsPage.com

In this section, SDO, where the mode stays in maintenance mode permanently once an initial organization of documents happens, and DDO will be evaluated and compared on the test bed, NewsPage.com. Two versions of the SDO system, 'SDO4KK' and 'SDO4NN', are used for this experiment and correspond to 'DDO4KK' and 'DDO4NN', respectively. In 'SDO4KK' and 'SDO4NN', their prefix, 'SDO' means SDO system, and the mean of '4KK' and '4NN' following the prefix, 'SDO' is same to that of 4KK' and '4NN' following the prefix, 'DDO'. For example, 'SDO4KK' is a version of SDO system of four-phase scenario where Kohonen Networks and K nearest neighbor are used as approaches to text clustering and text categorization, respectively. The parameters of machine learning algorithms involved in this experiment are set identically to those in the previous experiment. Like in the previous experiment, the clustering index after five steps of each manipulation is used to measure the quality of managing documents.

Figure 46 shows the result of evaluating SDO and DDO within four-phase scenario on the test bed, NewsPage.com. In figure 46, the left graph shows the performance of the two versions of SDO and the right graph shows the performance of the two versions of DDO. The two graphs in figure 46 show clearly that the two versions of the DDO system have much higher clustering indices in every manipulation.

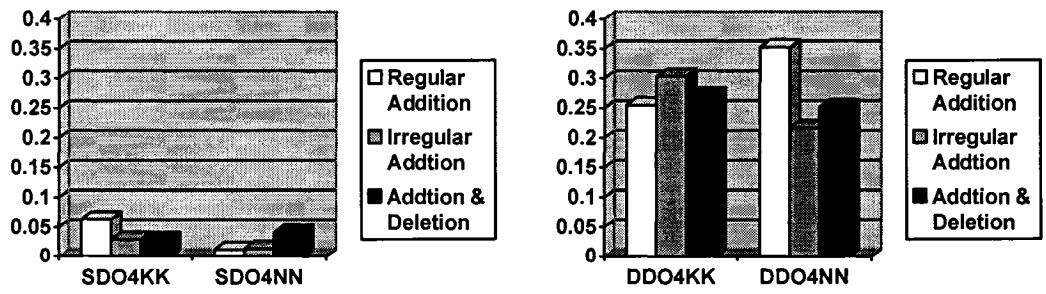

Figure 46. Clustering Index of SDO and DDO in NewsPage.com

In figure 47, the dominance of the two versions of the DDO system over those of the SDO system is visualized with a pie chart. In the pie chart, the white area corresponds to the mean clustering index of versions of DDO system, while the black area corresponds

to that of versions of SDO system. The mean clustering index of the versions of DDO system is 0.2751 while that of the versions of SDO system is 0.0314. Therefore, the results of this experiment set show clearly that there is the outstanding difference between the two paradigms of organizing documents on this test bed with respect to quality of managing documents.

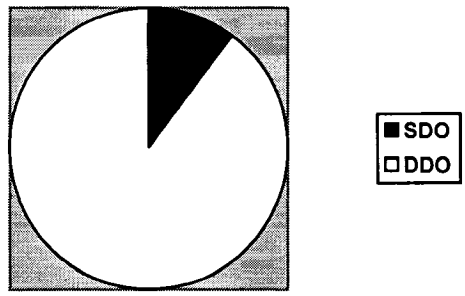

Figure 47. SDO vs DDO in NewsPage.com

6.4.2 20NewsGroups

In this section, SDO and DDO are evaluated and compared with each other on another test bed, '20NewsGroups'. Like the previous experiment, we use only four representative categories among 20 categories for evaluating versions of DDO and SDO.

Figure 48 shows the results of evaluating the two paradigms of document organization on the test bed, 20NewsGroups. The results show also that the versions of DDO system have much higher clustering indices in every manipulation defined in table 16 than those of SDO system.

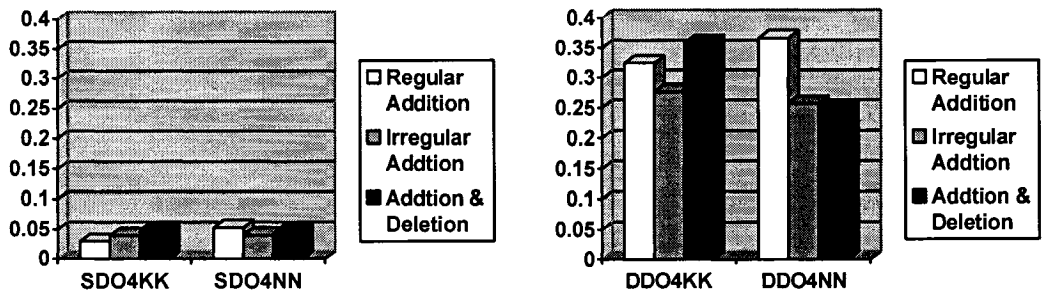

Figure 48. Clustering Index of SDO and DDO in 20NewsGroups

In figure 49, the dominance of the versions of DDO system over those of SDO system on this test bed is visualized. The mean clustering index of versions of DDO system over all of manipulations is 0.3052, while mean clustering index of versions of SDO system is 0.0425. This means that on this test bed, the two versions of DDO system manage documents with much higher quality eight times than those of SDO system.

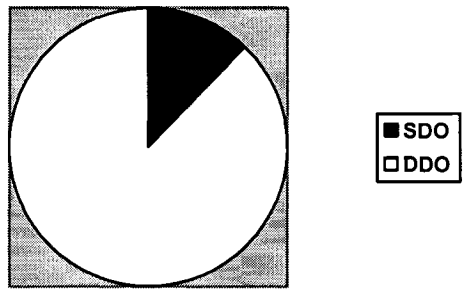

Figure 49. SDO vs DDO in 20NewsGroups

6.4.3 Discussion

The results of this experiment confirm that versions of DDO system are more desirable for managing documents than those of SDO system. The reason of generating the results is that versions of DDO system have more chance to improve quality of organizing documents by clustering them again. Versions of SDO have the risk of staying continually in a poor organization of documents, once documents are organized poorly in its initial creation mode. However, versions of DDO system are able to escape from a poor organization of documents by moving back to creation mode and reorganizing them, although it faces a very poor organization of documents.

6.5 String Vector vs Numerical Vector

This section displays the comparison of numerical vectors and string vectors as the representations of documents in implementing DDO systems with visualization. The experiments in chapter 4 concerning text clustering and text categorization showed NTC and NTSO using string vectors are comparable to the best traditional approaches using numerical vectors. Since the experiments in section 6.3 and 6.4 showed that four-phase scenario DDO is best way of implementing DDO systems, the comparison of the two representations of documents is within the best way: DDO4KK and DDO4NN.

Figure 50 visualizes the comparison of DDO4KK and DDO4NN in the first test bed, NewsPage.com. DDO4KK has its clustering index averaged over the three manipulations, 0.2768, while DDO4NN has its clustering index, 0.2763. As illustrated in figure 50, the two versions of the DDO systems are similar to each other with respect to the quality of managing documents. Note that DDO4NN uses smaller input size and iterations of learning process than DDO4KK.

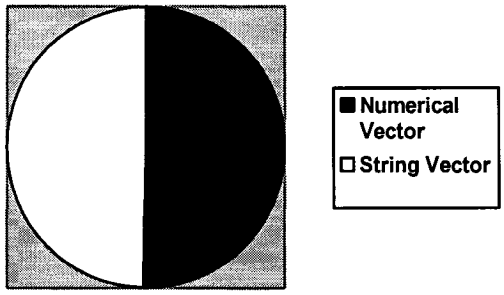

Figure 50. String vector vs Numerical Vector in NewsPage.com

Figure 51 visualizes the comparison of DDO4KK and DDO4NN in the second test bed, 20NewsGroups. DDO4KK has its averaged clustering index, 0.3199, while DDO4NN has its averaged clustering index, 0.2905. In this test bed, DDO4KK manages documents with slightly higher quality than DDO4NN. The version of the DDO system using string vectors is also comparable to the version using numerical vectors in this test bed.

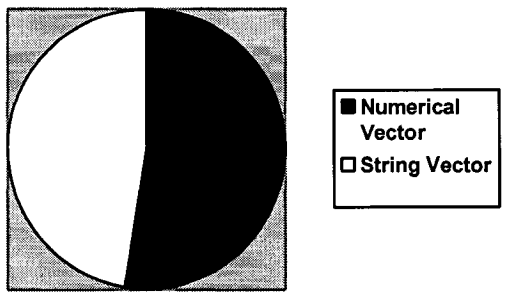

Figure 51. String Vector vs Numerical Vector in 20NewsGroups

These results show that string vectors are more practical representations of documents not only for text clustering and text categorization, but also for implementing DDO systems. The reason is that DDO4NN using string vectors is similar as DDO4KK with respect to quality of managing documents with the smaller input size and iterations of learning process. We can judge that DDO4NN is more advantageous than DDO4KK for implementing real time DDO systems.

6.6 Resampling vs Non Resampling

The last main experiment concerns the comparison of two versions of the DDO system, 'DDO4KK' and 'DDO4KKO'. The goal of this experiment is to observe the benefit of resampling for the performance of DDO system and determine whether a resampling method is adopted for developing DDO systems or not. Based on the results of the preliminary experiment in section 5.4, among resampling methods, mutate oversampling is selected for the version of DDO system, 'DDO4KKO' for this experiment.

Figure 52 and 53 show the results of evaluating the two versions of DDO system, 'DDO4KK' and 'DDO4KKO', on NewsPage.com and 20NewsGroups, respectively. In both figures, the white bars indicate clustering indices of DDO4KK which does not use

the oversampling, while the black bars indicate clustering indices of DDO4KKO which uses the oversampling. As illustrated in figure 52 and 53, it is shown that the oversampling method is not helpful for implementing DDO systems on both test beds.

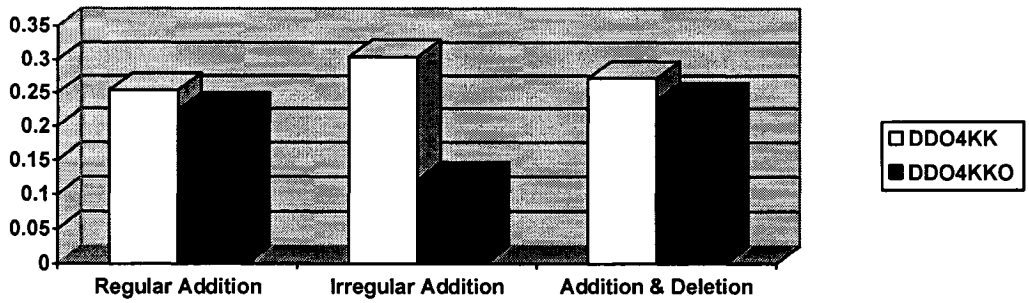

Figure 52. Non-resampling vs Resampling in DDO systems of four phases scenario on NewsPage.com

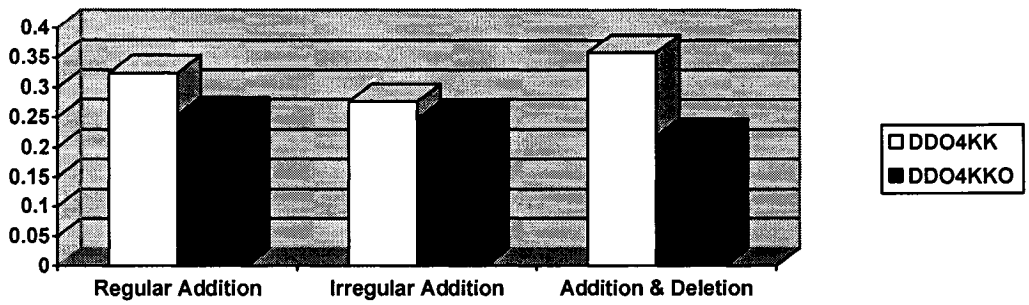

Figure 53. Non-resampling vs Resampling in DDO systems of four phases scenario on 20NewsGroups

The results of this experiment show that resampling methods are helpful only if training examples are prepared reliably: with little noise and overlapping. In versions of DDO system of four-phase scenario, documents organized in creation mode are used to train classifiers, instead of manually labeled documents. These documents may be labeled inconsistently with their target labels in creation mode when labeled documents are used for testing versions of DDO system, and they include noise and overlapping in their distribution. Since the phase, 'classifier training', of versions of DDO system uses training documents with such properties, resampling methods can not play their own function in this situation.

Chapter 7 Conclusion

This chapter, as the conclusion of this work, summarizes contents of this dissertation from chapter 1 to chapter 6, discusses main contributions of this research, and proposes areas for future research.

7.1 Summary

This section summarizes the content of this work from chapter 1 to chapter 6. Chapter 1 provided the background and the necessity of this research, as the introduction. Document organization, its paradigm, and its components necessary were defined for reading and understanding this material. Before describing what is proposed in this research, content based document organization is necessary for easy access and efficient management, whether it is manual or automatic. It was insisted that neither text categorization nor text clustering automates document organization completely, and that they should be combined with each other to do that completely. In chapter 1, the expected benefits of this research were presented; this research provides fundamental techniques for automating document organization completely for any textual information system and a way of avoiding the two problems in representing documents into numerical vectors by proposing the alternative representation.

In chapter 2, we reviewed previous works on the topics relevant to this research, the two existing representations of documents, and the existing approaches to text clustering and text categorization. Since documents consist of unstructured data which can not be processed directly by computers, the two representations of documents, bags of word and numerical vector, were described as a preprocessing step for text clustering and text categorization. We reviewed previous research on text clustering, the EM algorithm, and COBWEB which provided the basis for the idea of this research, and described the main traditional approaches to text clustering: the single pass algorithm, Kohonen Networks, and the EM algorithm. We reviewed cluster identification in previous research on text clustering, discovered that cluster identification was not considered except in the text clustering system, WEBSOM, and insisted that cluster identification is necessary for easy browsing. We also reviewed previous research on text categorization, and saw that labeled sample documents must be prepared for text categorization, whether unlabeled sample documents are used or not. We described the four common and traditional approaches to text categorization: K Nearest Neighbor, Naïve Bayes, Support Vector Machine, and Backpropagation. In the last section, we mentioned the class imbalance problem which usually happens when training classifiers and we reviewed previous research on resampling as solutions to the problem.

In chapter 3, we described DDO systems, which are proposed in this research, at the conceptual level. In the first section, we defined the concept of document organization before mentioning what is proposed, we discussed the two paradigms of document organization, DDO and SDO, and we proposed the two scenarios as the frames of implementing DDO systems. In the second section, we stated that text clustering, cluster identification, and text categorization are involved in implementing DDO systems, and we defined the roles of each system components. In the creation mode, text clustering generates unnamed clusters containing content based similar documents, and cluster identification names each cluster according to its content, based on the three conditions defined in this research. In the maintenance mode, text categorization trains a classifier or

classifiers with the documents that were organized in the creation mode, and arranges successive documents into their corresponding clusters by classifying them. We considered the conditions for transferring from the maintenance mode to the creation mode as the coordination of the three components.

In chapter 4, we proposed a better representation of documents, called string vector, in order to improve the performance of DDO systems. We described string vectors, the similarity matrix needed for computing them, and their advantages over the two traditional representations of documents. We described the two neural networks, NTSO and NTC, which use string vectors, instead of numerical vectors, as the recommended approaches to text clustering and text categorization. Although NTSO follows Kohonen Networks with respect to its architecture and learning rule, it uses string vectors as its input vectors and weight vectors, and performs the operations on string vectors, using a similarity matrix. NTC also uses string vectors. It follows Perceptrons in that its weight vectors are updated only when a training example is misclassified, and the value of each output node is a linear combination of weights. We compared the two neural networks with the traditional approaches to text clustering and text categorization described in chapter 2, and discovered that the proposed neural networks are competitive with the best traditional approach with the smaller input size and the smaller number of learning iterations.

In chapter 5, we described resampling methods as solutions for the class imbalance problem since the problem may be issued just before training a classifier or classifiers in DDO systems. We defined the role of resampling for DDO systems as an optional component, and discussed how to apply it to DDO systems. Spanning over the two sections, we described the oversampling methods and the undersampling method which are applicable to string vectors as well as to numerical vectors. We evaluated these resampling methods in text categorization, and discovered that they may be effective or not, depending on the approach. The experiment in section 5.4 presented that resampling methods are effective for K Nearest Neighbor and Naïve Bayes, but not for the other algorithms considered.

In chapter 6, we ran the main experiments on DDO systems, in order to evaluate the two scenarios of DDO systems, the two paradigms of document organization (DDO versus SDO), and the effectiveness of resampling on DDO systems. We also observed the effect of string vectors by comparing the two best versions of the DDO systems, DDO4KK and DDO4NN.. Before starting the experiments, we selected the versions of DDO systems participating in the experiments, based on the results of the experiments in section 4.4 and 5.4, and defined the measure used to evaluate DDO systems and SDO systems. We compared the two scenarios of DDO systems with each other, and determined that the four-phase scenario was more desirable for implementing DDO systems. We compared the two paradigms of document organization, and determined that DDO is more desirable for organizing documents than SDO. Observing the comparison of string vectors and numerical vectors for implementing DDO systems, DDO4NN is comparable to DDO4KK with the smaller input size and iterations of learning process. Therefore, string vectors is more practical than numerical vectors, as strategies of encoding documents for implementing DDO systems. We observed the effectiveness of resampling on DDO systems, by comparing the version with resampling with that without resampling, and determined that resampling is not effective on DDO systems.

7.2 Contributions of this Research

In this section, we discuss the significances of this research. The most important aspect of this research is the implementation of prototype programs which manage documents automatically. If these prototype programs were extended as independent systems or modules for textual information systems, they would eliminate time consuming jobs for managing documents in textual information systems; users or administrators would do nothing manually for managing documents other than adding and deleting them. Manual tasks involved in managing documents require not only very much time, but also background knowledge on the given domain; we must know the domain, in order to predefine categories and scan documents individually to arrange them. This research is expected to make it easier for users or administrators of textual information systems to manage documents.

This research proposed the fundamental framework of implementation of automatic document management systems. The frame consists of a creation mode and a maintenance mode. In the proposed frame, documents are managed automatically by switching between the two modes. In the creation mode, documents are organized initially or after time during operation, and in the maintenance mode, successive documents are inserted into the organization. This frame becomes a principle of implementing DDO systems.

This research proposes two specific scenarios specific for implementing DDO systems, based on the frame. In the three-phase scenario, text clustering and cluster identification are involved in the creation mode, and document classification is involved in the maintenance mode. In the four-phase scenario, the creation mode involves the same phases as in the three-phase scenario, and the maintenance mode involves classifier training as well as document classification. We implemented DDO systems based on the two scenarios as prototype programs, and evaluated them.

In this research, both text clustering and text categorization are integrated with cluster identification into the task of automatic document management. Previous research on text clustering and text categorization has progressed separately. Previous research has pursued more accurate approaches to either text categorization or text clustering by comparing them with other similar approaches, rather than by integrating them into automatic document management style for users and administrators. Text categorization or text clustering is insufficient, alone, to completely implement automatic document management. Similarly, text clustering generates unnamed clusters containing content based similar documents; these clusters are not sufficient for browsing. Text categorization requires, by itself, the manual preliminary tasks: predefinition of categories and preparation of sample labeled documents. This research shows that automatic document management can be implemented, if text clustering and text categorization are combined with cluster identification.

This research also proposed string vectors, an alternative representation of documents to bags of words and numerical vectors as a better representation for text categorization and text clustering. Almost all of machine learning algorithms require representing raw data into numerical vectors for applying them to any classification or clustering algorithm. Depending on a type of raw data, the representation may be difficult. For example, if raw data are documents, the representation leads to two problems: huge dimensionality and sparse distribution of numerical vectors. If the dimension of

numerical vectors is reduced using feature selection methods excessively, information loss takes the place of huge dimensionality. Even if SVM is very tolerant to huge dimensionalities, it does not address the sparse distribution problem. This research addresses the two problems at same time by proposing another representation of documents.

This research proposed the two neural networks, NTSO and NTC, which use string vectors, instead of numerical vectors. Although the two proposed neural networks follow traditional neural networks with respect to their architectures and learning rules, they are new models in that they use string vectors. The experimental results show that the two neural networks are comparable to the best traditional approach in terms of performance, while they have smaller input size and smaller number of learning iterations. More generally, the proposed representation may be useful in representing other types of raw data, such as images, proteins, and web documents, the proposed neural networks may be practical and useful for web mining, image processing, and bioinformatics, as well as text mining.

This research defined the principles of cluster identification, and proposed its process for browsing. Users usually prefer to access documents by browsing, rather than by searching. Cluster identifiers are necessary to decide whether a user access a given cluster or not, in advance, by reflecting on their contents, briefly. In order to accomplish this, we need the principles of identifying clusters based on their contents.

Finally, this research considered how to resample string vectors as well as numerical vectors. Before this research, resampling methods had been proposed only for numerical vectors. All of the resampling methods described in chapter 5 were modified for both numerical vectors and string vectors.

7.3 Future Research

In this research, we proposed and implemented DDO systems as prototype programs using techniques of text clustering and text categorization, together with cluster identification. It proposed string vectors as the alternative representation of documents and the two neural networks using the representation. However, we need further research on this topic for implementing DDO systems as real time systems or commercial systems. This section discusses the directions for future research.

7.3.1 Coordination of the Three Components

The coordination of the three components involved in implementing DDO systems is as important as the approaches to individual components. For example, in the current version of DDO systems, the transition from the maintenance mode and to the creation mode happens when the current number of transactions is higher than the threshold given as a parameter. This parameter influences on the quality of managing documents. If DDO systems were implemented as distributed systems, the coordination of the three components would become more even important and critical. In this section, we consider future research on the coordination to improve DDO systems.

For the coordination of the three components, the following two questions are issued: "When should the initial organization of documents be started?" and "When should the status of DDO systems move from the maintenance mode back to the creation mode?". It is assumed that every textual information system is always empty, when it is installed

initially. When a particular number of documents are piled in a system, they need to be organized. We must decide in advance what number of documents should trigger their initial organization.

The second question makes the distinction between DDO systems and SDO systems. In the current version of DDO systems, the transaction based method, among the three methods discussed in section 3.3, was adopted as the condition for transferring from the maintenance mode to the creation mode. If DDO systems are implemented as real time systems or commercial systems, the condition needs to be more complicated and sophisticated. Therefore, it is necessary to find optimal conditions of the transition as future research.

Since each DDO system is implemented as a single program, it runs sequentially along the components. When a large volume of documents accumulates in textual information systems, it may take very much time to reorganize documents in the creation mode and train classifiers just before the maintenance mode. While DDO systems perform these tasks, users are not allowed to access the systems. For user's convenience, DDO systems need to be implemented as distributed systems where the components are independent programs: clients and servers. If we implement DDO systems that way, issues of coordination of the involved components become very complicated and critical. Therefore, research on the coordination of the components is necessary for developing DDO systems as distributed systems.

In the experiments in chapter 6, the number of clusters is fixed to evaluate DDO systems. In the real world, we must consider the number of clusters whenever the status of the systems moves back to the creation mode. In section 3.3, we present two methods of determining it automatically. In a further research, we will evaluate the systems in the flexible number of clusters.

7.3.2 String Vectors

Although we could have defined features of string vectors in various ways, we defined them in descending order of frequency of words in a document, for simplicity, in this research. We could, instead, have considered not only word frequency, but also other factors, posting information and linguistic properties of words, for defining features. After this research, it is necessary to compare features of string vectors with each other within the proposed neural networks for text clustering and text categorization.

String vectors were proposed as a strategy for encoding documents to address the two problems of numerical vectors, rather than as a mathematical theory. Unlike numerical vectors, string vectors have no mathematical foundation, yet; other operations than the ones involved in training NTSO are not defined, and algebraic theory is not established. The disadvantages of string vectors are that they are more restrictive in their operations than numerical vectors, and each element of a string vector can not be treated as a quantity. Therefore we need theoretical and mathematical research for string vectors as in the future in order to allow for more flexible operations.

7.3.3 The Improvement of the Three-phase scenario

The experimental results presented in section 6.3 show that the DDO systems in the three-phase scenario are far less robust than those in the four-phase scenario. The advantage of the three-phase scenario is that there is no delay between the creation mode

and the maintenance mode for training classifiers. For this reason, the three-phase scenario may be preferred to the four-phase scenario for implementing real time DDO systems. We need to improve the three-phase scenario to be competitive with the four-phase scenario.

A strategy for improving DDO systems in a the three-phase scenario is to arrange documents using multiple prototypes, instead of a single prototype, per cluster. There are two kinds of ways that can be used for generating multiple prototypes for each cluster, even if current clustering algorithms generate only one prototype per cluster. One kind of ways is after clustering documents, to select more entities as additional prototypes. The determining criteria for selecting them are left as future research. The other kind of ways is the modification of existing clustering algorithms to generate multiple prototypes per cluster. If we use fuzzy concept, the modification becomes possible.

7.3.4 Using Resampling Methods

The experimental results presented in sections 5.4 and 6.5 show that resampling methods are effective on simple text categorization, but not effective on DDO systems. This means that the quality of the training data should be sufficient for using resampling methods. Resampling can address class imbalance problem, but not noise in the training data. Since, in simple text categorization, manually labeled documents are provided as clean sample documents, resampling is effective on the task. In the practical world, completely clean sample documents can not be given as data sets for text categorization. In DDO systems, sample documents are generated automatically during the creation mode. It is not avoidable that the documents have more noise than sample documents provided manually.

Before using resampling methods for DDO systems, sample documents generated during the creation mode should be cleaned. Here, cleaning means removing representations of sample documents with much noise. It is necessary to define criteria for judging whether representations of sample documents have much noise or not. This was also left as future research.

7.3.5 Improving NTSO

A disadvantage of NTSO is that it depends on a similarity matrix to perform operations involved in its learning process on string vectors. Before using NTSO, it takes very much time and system resource to build a similarity matrix from a large corpus. If there is no change in the domains of the documents, a similarity matrix can be reused continually, once it is built; building a similarity matrix is independent for clustering documents using NTSO. Therefore, time for building a similarity matrix is not counted as clustering time in this research. However, if documents in a different domain are clustered, we must build a similarity matrix, again. The reason is that each element of a similarity matrix indicating a semantic similarity between two words is very dependent on a domain of the given corpus; it is determined by collocations of two words in a given corpus. Since it is very expensive to build a similarity matrix, we need to find strategies for computing string vectors without it as future research.

Bibliography

Ambroise, C. and Govaert, G., "Convergence of an EM-type algorithm for spatial clustering", Pattern Recognition Letters, Vol 19, No 10, 1998, pp919-927.

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

Aras, N., Oommen, B. J., Altinel, I. K., "The Kohonen Networks incorporating explicit statistics and its application to the travelling salesman problem", Neural Networks, Vol 12, No 9, 1999, pp1273-1284.

Aslam, J., Pelekhov, K., and Rus, D., "Generating, Visualizing, and Evaluating High-Quality Clusters for Information Organization", Technical Report PCS-TR97-319, Department of Computer Science, Dartmouth College, 1997.

Banerjee, A., Dhillon, I., Ghosh, J., and Sra S., "Generative model-based clustering of directional data", The Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp19-28.

Baumann, T., Germond, A. J., and Sieman, A. G., "Application of the Kohonen Networks to short-term load forecasting", The Proceedings of the Second International Forum on Applications of Neural Networks to Power Systems, 1993, pp407-412.

Bote, G., Vincent, P., Felix, M. A., and Solana, V. H., "Document Organization using Kohonen's Algorithm", Information Processing and Management, Vol 38, No 1, 2002, pp79-89.

Celeux, G. and Govaert, G., "A Classification EM algorithm for clustering and two stochastic versions", Computational Statistics & Data Analysis, Vol 14, No 3, 1992, pp315-332.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P., "SMOTE: Synthetic Minority Oversampling Techniques", The Proceedings of Knowledge based Computer Systems, 2000, pp47-57.

Chawla, N. V., Japkowicz, N., and Kolcz, "Editorial: Special Issue on Learning from Imbalanced Data Sets", ACM SIGKDD Explorations Newsletter, Vol 6, No1, 2004, pp1-6.

Cohen W., "Text Categorization and Relational Learning", The Proceedings of 12th International Conference on Machine Learning, 1995, pp124-132.

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

Dempster A. P., Laird, N. M., and Rubin, D. B., "Maximum Likelihood from Incomplete Data via EM algorithm", Journal of the Royal Statistics Society, Series B, Vol 39, No 1, 1977, pp1-38.

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

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

Emamian, V., Kaveh, M., and Tewfik, A. H., "Robust Clustering of Acoustic Emission Signals using the Kohonen Networks", The Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000, pp3891-3894.

Estabrooks, A., Jo. T, and Japkowicz, N., "A Multiple Resampling Method for Learning from Imbalanced Data Sets", Vol 28, No 1, Computational Intelligence, 2004, pp18-36.
Fisher, D, "Knowledge Acquisition Via Incremental Conceptual Clustering", Machine Learning, Vol 2, 1987, 139-172.
Goldberg, D. E., Genetic Algorithms: in Search Optimization & Machine Learning, Addison Wesley Publishing Company, 1989.
Goldszidt, M. and Sahami, M., "A Probabilistic Approach to Full Text Document Clustering", Technical Report ITAD-433-MS-98-044, SRI International, 1998.
Hatzivassiloglou, V., Gravano, L., and Maganti, A., "An Investigation of Linguistic Features and Clustering Algorithms for Topical Document Clustering", The Proceedings of 23rd SIGIR, 2000, pp224-231.
Haykin, S., Neural Networks: Comprehensive Foundation, Macmillan College Publishing Company, 1994.
Hearst, M., "Support Vector Machines", IEEE Intelligent Systems, Vol 13, No 4, 1998, pp18-28.
Jacobs, R. A., "Increased Rates of Convergence through Learning Rate Adaptation", Neural Networks, Vol 1, No 4, 1988, pp295-308.
Japkowicz, N., "Concept Learning in the Presence of Between-Class and Within-Class Imbalances, The Proceedings of 14th Conference of the Canadian Society for Computational Studies of Intelligence, 2001, pp67-77.
Jo, T, "NeuroTextCategorizer: A New Model of Neural Network for Text Categorization", The Proceedings of International Conference of Neural Information Processing 2000, 2000, pp280-285.
Jo, T, "Machine Learning based Approach to Text Categorization with Resampling Methods", The Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics, 2004, pp93-98.
Jo, T. and Japkowicz, N., "Class Imbalances versus Small Disjuncts", ACM SIGKDD Explorations Newsletter, Vol 6, No1, 2004, pp40-49.
Jo, T. and Japkowicz, N., "Text Clustering with NTSO", The Proceedings of IJCNN 2005, 2005, pp558-563.
Jo, T. and Seo, J., "Text Categorization Oriented Connectionist Model", The Proceedings of ICCPOL, 2001, pp65-68.
Joachims, T., "Text Categorization with Support Vector Machines: Learning with many Relevant Features", The Proceedings of 10th European Conference on Machine Learning, 1998, pp143-151.
Kaski, S., Honkela, T., Lagus, K., and Kohonen, T., "WEBSOM-Self Organizing Maps of Document Collections", Neurocomputing, Vol 21, 1998, pp101-117.
Kohonen, T., "Self Organized Formation of Topologically Correct Feature Maps, Biological Cybernetics, Vol 43, 1982, pp59-69.
Kohonen T., Kaski, S., Lagus, K., Salojarvi, J., Paatero, V., and Saarela, A., "Self Organization of a Massive Document Collection", IEEE Transaction on Neural Networks, Vol 11, No 3, 2000, pp574-585.
Malmstrom, K., Munday, L., and Sitte, J., "A Simple Robust Robotic Vision System using Kohonen Feature Mapping", The Proceedings of the Second International Conference on Intelligent Information System, 1994, pp135-139.
Mitchell, T. M., Machine Learning, McGraw-Hill, 1997.

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

Mostafa, J. and Lam, W., "Automatic Classification using Supervised Learning in a Medical Document filtering Application", Information Processing and Management, Vol 36, 2000, pp415-444.

Muller, K.-R. Mika, S. Ratsch, G. Tsuda, K. Scholkopf, B., "An Introduction to Kernel-Based Learning Algorithms", IEEE Transaction on Neural Networks, Vol 12, No 2, 2001, pp181-201.

Nigam, K. and McCallum, A. K., Thrun S., and Mitchell, T. M., "Text Classification from Labeled and Unlabeled Documents using EM", Machine Learning, Vol 39, No 2-3, 2000, pp1-34.

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

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

Ritter, H., Martinetz, T., and Schulten, K., Neural Computation and Self Organizing Maps, Addison Wesley, 1991.

Rosenblatt, F., "The Perceptron: A Probabilistic Model for Information Storage and Organization in Brain", Psychological Review, Vol 65, 1958, pp386-408.

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

Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Representations by back-propagating Errors", Nature (London), Vol 323, 1986, pp533-536.

Salton, G., Automatic Text Processing, Addison-Wesley Publishing Company, 1989.

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

Skarmeta, A. G., Bensaid, A., and Tazi, N., "Data Mining for Text Categorization with semi-supervised Agglomerative Hierarchical Clustering", International Journal of Intelligent Systems, Vol 15, 2000, pp633-646.

Tan, P., Steinbach M., and Kumar, V., Introduction to Data Mining, Addison Wesely, 2006.

Truney, P.D., "Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL", The Proceedings of 12th ECML, 2001, pp3-7.

Vapnik, V., Estimation of Dependencies Based on Empirical Data, Springer Verlag, 1982.

Vinokourov, A. and Girolami, M., "A Probabilistic Hierarchical Clustering Method for Organizing Collections of Text Documents", The Proceedings of 15th International Conference on Pattern Recognition, 2000, pp182-185.

Weiss, G. M., "Learning with Rare Cases and Small Disjuncts", The Proceedings of the 15th International Conference on Machine Learning, 1998, pp558-565.

Weiss, G. M., "The Effect of Small Disjuncts and Class Distribution on Decision Tree Learning", PhD Dissertation, Department of Computer Science, Rutgers University, 2003.

Wiener, E. D., "A Neural Network Approach to Topic Spotting in Text", The Thesis of Master's Thesis, 1995.

Wu, M., Fuller, M., and Wilkinson, R., "Using Clustering and Classification Approaches to Interactive Retrieval, Information processing and management, Vol 37, No 3, 2001, pp459-484.

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

Yao, X., "A Review of Evolutionary Artificial Neural Networks", International Journal of Intelligent System, Vol 4, 1993, pp203-222.