Nonhierarchical Document Clustering Based on a Tolerance Rough Set Model

Tu Bao Ho,[1*] Ngoc Binh Nguyen[2]
[1] *Japan Advanced Institute of Science and Technology, Tatsunokuchi, Ishikawa 923-1292, Japan*
[2] *Hanoi University of Technology, DaiCoViet Road, Hanoi, Vietnam*

Document clustering, the grouping of documents into several clusters, has been recognized as a means for improving efficiency and effectiveness of information retrieval and text mining. With the growing importance of electronic media for storing and exchanging large textual databases, document clustering becomes more significant. Hierarchical document clustering methods, having a dominant role in document clustering, seem inadequate for large document databases as the time and space requirements are typically of order $O(N^3)$ and $O(N^2)$, where N is the number of index terms in a database. In addition, when each document is characterized by only several terms or keywords, clustering algorithms often produce poor results as most similarity measures yield many zero values. In this article we introduce a nonhierarchical document clustering algorithm based on a proposed tolerance rough set model (TRSM). This algorithm contributes two considerable features: (1) it can be applied to large document databases, as the time and space requirements are of order $O(N\log N)$ and $O(N)$, respectively; and (2) it can be well adapted to documents characterized by a few terms due to the TRSM's ability of semantic calculation. The algorithm has been evaluated and validated by experiments on test collections.

1. INTRODUCTION

With the growing importance of electronic media for storing and exchanging textual information, there is an increasing interest in methods and tools that can help find and sort information included in the text documents.[4] It is known that *document clustering*—the grouping of documents into clusters—plays a significant role in improving efficiency, and can also improve effectiveness of text retrieval as it allows cluster-based retrieval instead of full retrieval. Document clustering is a difficult clustering problem for a number of reasons,[3,7,19] and some problems occur additionally when doing clustering on large textual databases. Particularly, when each document in a large textual database is represented by only a few keywords, current available similarity measures in textual clustering[1,3] often yield zero values

*Author to whom all correspondence should be addressed.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 17, 199–212 (2002)

that considerably decreases the clustering quality. Although having a dominant role in document clustering,[19] hierarchical clustering methods seem not to be appropriate for large textual databases, as they typically require computational time and space of order $O(N^3)$ and $O(N^2)$, respectively, where N is the total number of terms in a textual database. In such a case, nonhierarchical clustering methods are better adapted, as their computational time and space requirements are much less.[7]

Rough set theory, a mathematical tool to deal with vagueness and uncertainty introduced by Pawlak in the early 1980s,[10] has been successful in many applications.[8,11] In this theory each set in a universe is described by a pair of ordinary sets called lower and upper approximations, determined by an equivalence relation in the universe. The use of the original rough set model in information retrieval, called the *equivalence rough set model* (ERSM), has been investigated by several researchers.[12,16] A significant contribution of ERSM to information retrieval is that it suggested a new way to calculate the semantic relationship of words based on an organization of the vocabulary into equivalence classes. However, as analyzed in Ref. 5, ERSM is not suitable for information retrieval due to the fact that the requirement of the transitive property in equivalence relations is too strict the meaning of words, and there is no way to automatically calculate equivalence classes of terms. Inspired by some works that employ different relations to generalize new models of rough set theory, for example, Refs. 14 and 15 a *tolerance rough set model* (TRSM) for information retrieval that adopts tolerance classes instead of equivalence classes has been developed.[5]

In this article we introduce a TRSM-based nonhierarchical clustering algorithm for documents. The algorithm can be applied to large document databases as the time and space requirements are of order $O(N\log N)$ and $O(N)$, respectively. It can also be well adapted to cases where each document is characterized by only a few index terms or keywords, as the use of upper approximations of documents makes it possible to exploit the semantic relationship between index terms. After a brief recall of the basic notions of document clustering and the tolerance rough set model in Section 2, we will present in Section 3 how to determine tolerance spaces and the TRSM nonhierarchical clustering algorithm. In Section 4 we report experiments with five test collections for evaluating and validating the algorithm on clustering tendency and stability, efficiency, and effectiveness of cluster-based information retrieval in contrast to full retrieval.

2. PRELIMINARIES

2.1. Document Clustering

Consider a set of documents $\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$ where each document d_j is represented by a set of index terms t_i (for example, keywords) each is associated with a weight $w_{ij} \in [0, 1]$ that reflects the importance of t_i in d_j, that is, $d_j = (t_{1j}, w_{1j}; t_{2j}, w_{2j}; \ldots; t_{rj}, w_{rj})$. The set of all index terms from $\mathcal{D}$ is denoted by $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$. Given a query in the form $Q = (q_1, w_{1q}; q_2, w_{2q}; \ldots; q_s, w_{sq})$ where $q_i \in \mathcal{T}$ and $w_{iq} \in [0, 1]$, the *information retrieval* task can be viewed as to find ordered documents $d_j \in \mathcal{D}$ that are relevant to the query Q.

A *full search* strategy examines the whole document set $\mathcal{D}$ to find relevant documents of Q. If the document set $\mathcal{D}$ can be divided into clusters of related documents,

the *cluster-based search* strategy can considerably increase retrieval efficiency as well as retrieval effectiveness by searching the answer only in appropriate clusters. The *hierarchical* clustering of documents has been largely considered.[2,6,18,19] However, with the typical time and space requirements of order $O(N^3)$ and $O(N^2)$, hierarchical clustering is not suitable for large collections of documents. *Nonhierarchical* clustering techniques, with their costs of order $O(N\log N)$ and $O(N)$, certainly are much more adequate for large document databases.[7] Most nonhierarchical clustering methods produce partitions of documents. However, according to the overlapping meaning of words, nonhierarchical clustering methods that produce overlapping document classes serve to improve the retrieval effectiveness.

2.2. Tolerance Rough Set Model

The starting point of rough set theory is that each set X in a universe U can be "viewed" approximately by its upper and lower approximations in an approximation space $\mathcal{R} = (U, R)$, where $R \subseteq U \times U$ is an equivalence relation. Two objects $x, y \in U$ are said to be indiscernible regarding R if xRy. The *lower* and *upper approximations* in $\mathcal{R}$ of any $X \subseteq U$, denoted respectively by $\mathcal{L}(\mathcal{R}, X)$ and $\mathcal{U}(\mathcal{R}, X)$, are defined by

$$\mathcal{L}(\mathcal{R}, X) = \{x \in U : [x]_R \subseteq X\} \tag{1}$$

$$\mathcal{U}(\mathcal{R}, X) = \{x \in U : [x]_R \cap X \neq \phi\} \tag{2}$$

where $[x]_R$ denotes the equivalence class of objects indiscernible with x regarding the equivalence relation R. All early work on information retrieval using rough sets was based on ERSM with a basic assumption that the set $\mathcal{T}$ of index terms can be divided into equivalence classes determined by equivalence relations.[12,16] In our observation among the three properties of an equivalence relation R (reflexive, xRx; symmetric, $xRy \rightarrow yRx$; and transitive, $xRy \wedge yRz \rightarrow xRz$ for $\forall x, y, z \in U$), the transitive property does not always hold in certain application domains, particularly in natural language processing and information retrieval. This remark can be illustrated by considering words from Roget's thesaurus, where each word is associated with a class of other words that have similar meanings. Figure 1 shows associated classes of three words, *root, cause,* and *basis*. It is clear that these classes are not disjoint (equivalence classes), but overlapping, and the meaning of the words is not transitive.

Overlapping classes can be generated by *tolerance relations* that require only reflexive and symmetric properties. A general approximation model using tolerance relations was introduced in Ref. 14 in which generalized spaces are called *tolerance spaces* that contain overlapping classes of objects in the universe (tolerance classes). In Ref. 14, a tolerance space is formally defined as a quadruple $\mathcal{R} = (U, I, \nu, P)$, where U is a universe of objects, $I : U \rightarrow 2^U$ is an uncertainty function, $\nu : 2^U \times 2^U \rightarrow [0, 1]$ is a vague inclusion, and $P : I(U) \rightarrow \{0, 1\}$ is a structurality function.

We assume that an object x is perceived by information $\mathrm{Inf}(x)$ about it. The uncertainty function $I : U \rightarrow 2^U$ determines $I(x)$ as a tolerance class of all objects that are considered to have *similar* information to x. This uncertainty function can be any function satisfying the condition $x \in I(x)$ and $y \in I(x)$ iff $x \in I(y)$ for any

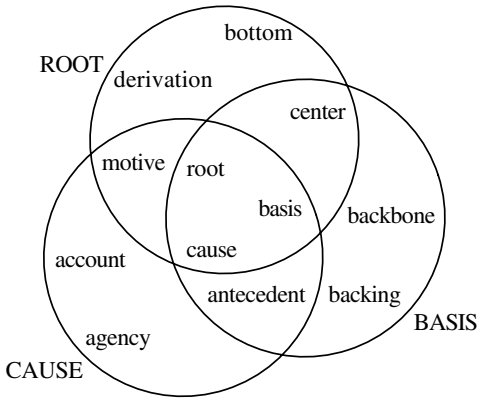

Figure 1. Overlapping classes of words.

$x, y \in U$. Such a function corresponds to a relation $\mathcal{I} \subseteq U \times U$ understood as $x\mathcal{I}y$ iff $y \in I(x)$. $\mathcal{I}$ is a tolerance relation because it satisfies the properties of reflexivity and symmetry.

The vague inclusion $\nu : 2^U \times 2^U \to [0, 1]$ measures the degree of inclusion of sets; in particular it relates to the question of whether the tolerance class $I(x)$ of an object $x \in U$ is included in a set X. There is only one requirement of *monotonicity* with respect to the second argument of ν, that is, $\nu(X, Y) \leq \nu(X, Z)$ for any $X, Y, Z \subseteq U$ and $Y \subseteq Z$.

Finally, the structurality function is introduced by analogy with mathematical morphology.[14] In the construction of the lower and upper approximations, only tolerance sets being structural elements are considered. We define that $P : I(U) \to \{0, 1\}$ classifies $I(x)$ for each $x \in U$ into two classes—structural subsets ($P(I(x)) = 1$) and non-structural subsets ($P(I(x)) = 0$). The lower approximation $\mathcal{L}(\mathcal{R}, X)$ and the upper approximation $\mathcal{U}(\mathcal{R}, X)$ in $\mathcal{R}$ of any $X \subseteq U$ are defined as

$$\mathcal{L}(\mathcal{R}, X) = \{x \in U \mid P(I(x)) = 1 \;\&\; \nu(I(x), X) = 1\} \tag{3}$$

$$\mathcal{U}(\mathcal{R}, X) = \{x \in U \mid P(I(x)) = 1 \;\&\; \nu(I(x), X) > 0\} \tag{4}$$

The basic problem of using tolerance spaces in any application is how to determine suitably I, ν, and P.

3. TRSM NONHIERARCHICAL CLUSTERING

3.1. Determination of Tolerance Spaces

We first describe how to determine suitably I, ν, and P for the information retrieval problem. First of all, to define a tolerance space $\mathcal{R}$, we choose the universe U as the set $\mathcal{T}$ of all index terms

$$U = \{t_1, t_2, \ldots, t_N\} = \mathcal{T} \tag{5}$$

The most crucial issue in formulating a TRSM for information retrieval is identification of tolerance classes of index terms. There are several ways to identify conceptually similar index terms, for example, human experts, thesaurus, term co-occurrence, and so on. We employ the co-occurrence of index terms in all documents from $\mathcal{D}$ to determine a tolerance relation and tolerance classes. The co-occurrence of index terms is chosen for the following reasons: (1) it gives a meaningful interpretation in the context of information retrieval about the dependency and the semantic relation of index terms[17]; and (2) it is relatively simple and computationally efficient. Note that the co-occurrence of index terms is not transitive and cannot be used automatically to identify equivalence classes. Denote by $f_{\mathcal{D}}(t_i, t_j)$ the number of documents in $\mathcal{D}$ in which two index terms t_i and t_j co-occur. We define the uncertainty function I depending on a threshold θ as

$$I_\theta(t_i) = \{t_j \mid f_{\mathcal{D}}(t_i, t_j) \geq \theta\} \cup \{t_i\} \tag{6}$$

It is clear that the function I_θ defined above satisfies the condition of $t_i \in I_\theta(t_i)$ and $t_j \in I_\theta(t_i)$ iff $t_i \in I_\theta(t_j)$ for any $t_i, t_j \in \mathcal{T}$, and so I_θ is both reflexive and symmetric. This function corresponds to a tolerance relation $\mathcal{I} \subseteq \mathcal{T} \times \mathcal{T}$ that $t_i \mathcal{I} t_j$ iff $t_j \in I_\theta(t_i)$, and $I_\theta(t_i)$ is the tolerance class of index term t_i. The vague inclusion function ν is defined as

$$\nu(X, Y) \;=\; \frac{|X \cap Y|}{|X|} \tag{7}$$

This function is clearly monotonous with respect to the second argument. Based on this function ν, the membership function μ for $t_i \in \mathcal{T}$, $X \subseteq \mathcal{T}$ can be defined as

$$\mu(t_i, X) = \nu(I_\theta(t_i), X) \;=\; \frac{|I_\theta(t_i) \cap X|}{|I_\theta(t_i)|} \tag{8}$$

Suppose that the universe $\mathcal{T}$ is closed during the retrieval process; that is, the query Q consists of only terms from $\mathcal{T}$. Under this assumption we can consider all tolerance classes of index terms as structural subsets; that is, $P(I_\theta(t_i)) = 1$ for any $t_i \in \mathcal{T}$. With these definitions we obtained the tolerance space $\mathcal{R} = (\mathcal{T}, I, \nu, P)$ in which the *lower approximation* $\mathcal{L}(\mathcal{R}, X)$ and the *upper approximation* $\mathcal{U}(\mathcal{R}, X)$ in $\mathcal{R}$ of any subset $X \subseteq \mathcal{T}$ can be defined as

$$\mathcal{L}(\mathcal{R}, X) = \{t_i \in \mathcal{T} \mid \nu(I_\theta(t_i), X) = 1\} \tag{9}$$

$$\mathcal{U}(\mathcal{R}, X) = \{t_i \in \mathcal{T} \mid \nu(I_\theta(t_i), X) > 0\} \tag{10}$$

Denote by $f_{d_j}(t_i)$ the number of occurrences of term t_i in d_j (term frequency), and by $f_{\mathcal{D}}(t_i)$ the number of documents in $\mathcal{D}$ that term t_i occurs in (document frequency). The weights w_{ij} of terms t_i in documents d_j is defined as follows. They are first calculated by

$$w_{ij} = \begin{cases} (1 + \log(f_{d_j}(t_i))) \times \log \frac{M}{f_{\mathcal{D}}(t_i)} & \text{if } t_i \in d_j, \\ 0 & \text{if } t_i \notin d_j \end{cases} \tag{11}$$

then are normalized by vector length as $w_{ij} \leftarrow w_{ij} / \sqrt{\sum_{t_h \in d_j} (w_{hj})^2}$. This

term-weighting method is extended to define weights for terms in the upper approximation $\mathcal{U}(\mathcal{R}, d_j)$ of d_j. It ensures that each term in the upper approximation of d_j, but not in d_j, has a weight smaller than the weight of any term in d_j:

$$w_{ij} = \begin{cases} (1 + \log(f_{d_j}(t_i))) \times \log \frac{M}{f_{\mathcal{D}}(t_i)} & \text{if } t_i \in d_j, \\ \min_{t_h \in d_j} w_{hj} \times \frac{\log(M/f_{\mathcal{D}}(t_i))}{1+\log(M/f_{\mathcal{D}}(t_i))} & \text{if } t_i \in \mathcal{U}(\mathcal{R}, d_j) \backslash d_j \\ 0 & \text{if } t_i \notin \mathcal{U}(\mathcal{R}, d_j) \end{cases} \tag{12}$$

The vector length normalization is then applied to the upper approximation $\mathcal{U}(\mathcal{R}, d_j)$ of d_j. Note that the normalization is done when considering a given set of index terms.

We illustrate the notions of TRSM by using the JSAI database of articles and papers of the Journal of the Japanese Society for Artificial Intelligence (JSAI) after its first ten years of publication (1986–1995). The JSAI database consists of 802 documents. In total, there are 1,823 keywords in the database, and each document has on average five keywords. To illustrate the introduced notions, let us consider a part of this database that consists of the first ten documents concerning "machine learning." The keywords in this small universe are indexed by their order of appearance, that is, $t_1 =$ "machine learning," $t_2 =$ "knowledge acquisition", ..., $t_{30} =$ "neural networks," $t_{31} =$ "logic programming." With $\theta = 2$, by definition (See Equation 6) we have tolerance classes of index terms $I_2(t_1) = \{t_1, t_2, t_5, t_{16}\}$, $I_2(t_2) = \{t_1, t_2, t_4, t_5, t_{26}\}$, $I_2(t_4) = \{t_2, t_4\}$, $I_2(t_5) = \{t_1, t_2, t_5\}$, $I_2(t_6) = \{t_6, t_7\}$, $I_2(t_7) = \{t_6, t_7\}$, $I_2(t_{16}) = \{t_1, t_{16}\}$, $I_2(t_{26}) = \{t_2, t_{26}\}$, and each of the other index terms has the corresponding tolerance class consisting of only itself, for example, $I_2(t_3) = \{t_3\}$. Table I shows these ten documents, and their lower and upper approximations with $\theta = 2$.

3.2. TRSM Nonhierarchical Clustering Algorithm

Table II describes the TRSM nonhierarchical clustering algorithm. It can be considered as a reallocation clustering method to form K clusters of a collection $\mathcal{D}$ of M documents.[3] The distinction of the TRSM nonhierarchical clustering

Table I. Approximations of first 10 documents concerning "machine learning."

	Keywords	$\mathcal{L}(\mathcal{R}, d_j)$	$\mathcal{U}(\mathcal{R}, d_j)$
d_1	t_1, t_2, t_3, t_4, t_5	t_3, t_4, t_5	$t_1, t_2, t_3, t_4, t_5, t_{16}, t_{26}$
d_2	t_6, t_7, t_8, t_9	t_6, t_7, t_8, t_9	t_6, t_7, t_8, t_9
d_3	$t_5, t_1, t_{10}, t_{11}, t_2$	t_5, t_{10}, t_{11}	$t_1, t_2, t_4, t_5, t_{10}, t_{11}, t_{16}, t_{26}$
d_4	$t_6, t_7, t_{12}, t_{13}, t_{14}$	$t_6, t_7, t_{12}, t_{13}, t_{14}$	$t_6, t_7, t_{12}, t_{13}, t_{14}$
d_5	t_2, t_{15}, t_4	t_4, t_{15}	$t_1, t_2, t_4, t_5, t_{15}, t_{26}$
d_6	$t_1, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}$	$t_{16}, t_{17}, t_{18}, t_{19}, t_{20}$	$t_1, t_2, t_5, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}$
d_7	$t_{21}, t_{22}, t_{23}, t_{24}, t_{25}$	$t_{21}, t_{22}, t_{23}, t_{24}, t_{25}$	$t_{21}, t_{22}, t_{23}, t_{24}, t_{25}$
d_8	$t_2, t_{12}, t_{26}, t_{27}$	t_{12}, t_{26}, t_{27}	$t_1, t_2, t_4, t_5, t_{12}, t_{26}, t_{27}$
d_9	t_{26}, t_2, t_{28}	t_{26}, t_{28}	$t_1, t_2, t_4, t_5, t_{26}, t_{28}$
d_{10}	$t_1, t_{16}, t_{21}, t_{26}, t_{29}, t_{30}, t_{31}$	$t_{16}, t_{21}, t_{26}, t_{29}, t_{30}, t_{31}$	$t_1, t_2, t_5, t_{16}, t_{21}, t_{26}, t_{29}, t_{30}, t_{31}$

Table II. The TRSM nonhierarchical clustering algorithm.

Input	The set $\mathcal{D}$ of documents and the number K of clusters
Result	K overlapping clusters of $\mathcal{D}$ associated with cluster membership of each document

1. Determine the initial representatives $R_1, R_2, \ldots, R_K$ of clusters $C_1, C_2, \ldots, C_K$ as K randomly selected documents in $\mathcal{D}$.
2. For each $d_j \in \mathcal{D}$, calculate the similarity $S(\mathcal{U}(\mathcal{R}, d_j), R_k)$ between its upper approximation $\mathcal{U}(\mathcal{R}, d_j)$ and the cluster representative R_k, for $k = 1, \ldots, K$. If this similarity is greater than a given threshold, assign d_j to C_k and take this similarity value as the cluster membership $m(d_j)$ of d_j in C_k.
3. For each cluster C_k, re-determine its representative R_k.
4. Repeat steps 2 and 3 until there is little or no change in cluster membership during a pass through $\mathcal{D}$.
5. Denote by d_u an unclassified document after steps 2, 3, and 4, and by $\mathrm{NN}(d_u)$ its nearest neighbor document (with non-zero similarity) in formed clusters. Assign d_u into the cluster that contains $\mathrm{NN}(d_u)$, and determine the cluster membership of d_u in this cluster as the product $m(d_u) = m(\mathrm{NN}(d_u)) \times S(\mathcal{U}(\mathcal{R}, d_u), \mathcal{U}(\mathcal{R}, \mathrm{NN}(d_u)))$. Re-determine the representatives R_k, for $k = 1, \ldots, K$.

algorithm is that it forms overlapping clusters and uses approximations of documents and cluster's representatives in calculating their similarity. The latter allows us to find some semantic relatedness between documents even when they do not share common index terms. After determining initial cluster representatives in step 1, the algorithm mainly consists of two phases. The first does an iterative re-allocation of documents into overlapping clusters by steps 2, 3, and 4. The second does, by step 5, an assignment of documents, that are not classified in the first phase, into clusters containing their nearest neighbors with non-zero similarity. Two important issues of the algorithms will be further considered: (1) how to define the representatives of clusters; and (2) how to determine the similarity between documents and the cluster representatives.

3.2.1. Representatives of Clusters

The TRSM clustering algorithm constructs a *polythetic* representative R_k for each cluster C_k, $k = 1, \ldots, K$. In fact, R_k is a set of index terms such that:

- Each document $d_j \in C_k$ has some or many terms in common with R_k
- Terms in R_k are possessed by a large number of $d_j \in C_k$
- No term in R_k must be possessed by every document in C_k

It is well known in Bayesian learning that the decision rule with minimum error rate to assign a document d_j in the cluster C_k is

$$P(d_j \mid C_k)P(C_k) > P(d_j \mid C_h)P(C_h), \quad \forall h \neq k \tag{13}$$

When it is assumed that the terms occur independently in the documents, we have

$$P(d_j \mid C_k) = P(t_{j_1} \mid C_k)P(t_{j_2} \mid C_k) \ldots P(t_{j_p} \mid C_k) \tag{14}$$

Denote by $f_{C_k}(t_i)$ the number of documents in C_k that contain t_i; we have $P(t_i \mid C_k) = f_{C_k}(t_i)/|C_k|$. In step 3 of the algorithm, all terms occurring in documents belonging to C_k in step 2 will be considered to add to R_k, and all terms existing in R_k will be considered to remove from or to remain in R_k. Equation 14 and heuristics of the polythetic properties of the cluster representatives lead us to adopt rules to form the cluster representatives:

(1) Initially, $R_k = \phi$
(2) For all $d_j \in C_k$ and for all $t_i \in d_j$, if $f_{C_k}(t_i)/|C_k| > \sigma$, then $R_k = R_k \cup \{t_i\}$
(3) If $d_j \in C_k$ and $d_j \cap R_k = \phi$, then $R_k = R_k \cup \text{argmax}_{t_i \in d_j} w_{ij}$

The weights of terms t_i in R_k are first averaged by weights of terms in all documents belonging to C_k, that means $w_{ik} = (\sum_{d_j \in C_k} w_{ij})/|\{d_j : t_i \in d_j\}|$, then normalized by the length of the representative R_k.

3.2.2. Similarity between Documents and the Cluster Representatives

Many similarity measures between documents can be used in the TRSM clustering algorithm. Three common coefficients of Dice, Jaccard, and Cosine[1,3] are implemented in the TRSM clustering program to calculate the similarity between pairs of documents d_{j_1} and d_{j_2}. For example, the Dice coefficient is

$$S_D(d_{j_1}, d_{j_2}) = \frac{2 \times \sum_{k=1}^{N} (w_{kj_1} \times w_{kj_2})}{\sum_{k=1}^{N} w_{kj_1}^2 + \sum_{k=1}^{N} w_{kj_2}^2} \tag{15}$$

When binary term weights are used, this coefficient is reduced to

$$S_D(d_{j_1}, d_{j_2}) = \frac{2 \times C}{A + B} \tag{16}$$

where C is the number of terms that d_{j_1} and d_{j_2} have in common, and A and B are the number of terms in d_{j_1} and d_{j_2}. It is worth noting that the Dice coefficient (or any other well-known similarity coefficient used for documents[1,3]) yields a large number of zero values when documents are represented by only a few terms, as many of them may have no terms in common ($C = 0$). The use of the tolerance upper approximation of documents and of the cluster representatives allows the TRSM algorithm to improve this situation. In fact, in the TRSM clustering algorithm, the normalized Dice coefficient is applied to the upper approximation of documents $\mathcal{U}(\mathcal{R}, d_j)$; that is, $S_D(\mathcal{U}(\mathcal{R}, d_j), R_k))$ is used in the algorithm instead of $S_D(d_j, R_k)$. Two main advantages of using upper approximations are:

(1) To reduce the number of zero-valued coefficients by considering documents themselves together with the related terms in tolerance classes.
(2) The upper approximations formed by tolerance classes make it possible to retrieve documents that may have few (or even no) terms in common with the query.

Table III. Test collections.

Collection	Subject	Documents	Queries	Relevant
JSAI	Artificial Intelligence	802	20	32
CACM	Computer Science	3,200	64	15
CISI	Library Science	1,460	76	40
CRAN	Aeronautics	1,400	225	8
MED	Medicine	3,078	30	23

4. VALIDATION AND EVALUATION

We report experiment results on clustering tendency and stability, as well as on cluster-based retrieval effectiveness and efficiency.[3,19] Table III summarizes test collections used in our experiments, including JSAI where each document is represented on average by five keywords, and four other common test collections.[3] Columns 3, 4, and 5 show the number of documents, queries, and the average number of relevant documents for queries. The clustering quality for each test collection depends on parameter θ in the TRSM and on σ in the clustering algorithm. We can note that the higher value of θ, the larger the upper approximation and the smaller the lower approximation of a set X. Our experiments suggested that when the average number of terms in documents is high and/or the size of the document collection is large, high values of θ are often appropriate and vice versa. In Table VI of Section 4.3, we can see how retrieval effectiveness relates to different values of θ. To avoid biased experiments when comparing algorithms, we take default values $K = 15$, $\theta = 15$, and $\sigma = 0.1$ for all five test collections. Note that the TRSM nonhierarchical clustering algorithm yields at most 15 clusters, as in some cases several initial clusters can be merged into one during the iteration process, and for $\theta \geq 6$, upper approximations of terms in JSAI become stable (unchanged).

4.1. Validation of Clustering Tendency

The experiments attempt to determine whether worthwhile retrieval performance would be achieved by clustering a database, before investing the computational resources that clustering the database would entail.[3] We employ the *nearest neighbor test*[19] by considering, for each relevant document of a query, how many of its n nearest neighbors are also relevant, and by averaging over all relevant documents for all queries in a test collection in order to obtain single indicators. We use in these experiments five test collections with all queries and their relevant documents.

The experiments are carried out to calculate the percentage of relevant documents in the database that had zero, one, two, three, four, or five relevant documents in the set of five nearest neighbors of each relevant document. Table IV reports the experimental results synthesized from those done on five test collections. Columns 2 and 3 show the number of queries and total number of relevant documents for all queries in each test collection. The next six rows show the average percentage of the relevant documents in a collection that had zero, one, two, three, four, and five relevant documents in their sets of five nearest neighbors. For example, the meaning of row JSAI column 9 is "among all relevant documents for 20 queries of the JSAI

Table IV. Results of clustering tendency.

	Queries	# Relevant documents	% average of relevant documents						Average
			0	1	2	3	4	5	
JSAI	20	32	19.9	19.8	18.5	18.5	11.8	11.5	2.2
CACM	64	15	50.3	22.5	12.8	7.9	4.2	2.3	1.0
CISI	76	40	45.4	25.8	15.0	7.5	4.3	1.9	1.1
CRAN	225	8	33.4	32.7	19.2	9.0	4.6	1.0	1.2
MED	30	23	10.4	18.7	18.6	21.6	19.6	11.1	2.5

collection, 11.5 percent of them have five nearest neighbor documents all as relevant documents." The last column shows the average number of relevant documents among five nearest neighbors of each relevant document. This value is relatively high for the JSAI and MED collections and relatively low for the others.

As the finding of nearest neighbors of a document in this method is based on the similarity between the upper approximations of documents, this tendency suggests that the TRSM clustering method might appropriately be applied for retrieval purposes. This tendency can be clearly observed in concordance with the high retrieval effectiveness for the JSAI and MED collections shown in Table VI.

4.2. The Stability of Clustering

The experiments were done for the JSAI test collection in order to validate the stability of the TRSM clustering, that is, to verify whether the TRSM clustering method produces a clustering that is unlikely to be altered drastically when further documents are incorporated. For each value 2, 3, and 4 of θ, the experiments are done ten times each for a reduced database of size $(100 - s)$ percent of $\mathcal{D}$. We randomly remove a specified of s percentage documents from the JSAI database, then re-determine the new tolerance space for the reduced database. Once having the new tolerance space, we perform the TRSM clustering algorithm and evaluate the change of clusters due to the change of the database. Table V synthesizes the experimental results with different values of s from 210 experiments with $s = 1$, 2, 3, 4, 5, 10, and 15 percent.

Note that a little change of data implies a possible little change of clustering (about the same percentage as for $\theta = 4$). The experiments on the stability for other test collections have nearly the same results as those of the JSAI. That suggests that the TRSM nonhierarchical clustering method is highly stable.

Table V. Synthesized results about the stability.

	Percentage of changed data						
	1%	2%	3%	4%	5%	10%	15%
$\theta = 2$	2.84	5.62	7.20	5.66	5.48	11.26	14.41
$\theta = 3$	3.55	4.64	4.51	6.33	7.93	12.06	15.85
$\theta = 4$	0.97	2.65	2.74	4.22	5.62	8.02	13.78

Table VI. Precision and recall of full retrieval.

	JSAI		CACM		CISI		CRAN		MED	
θ	P	R	P	R	P	R	P	R	P	R
30	0.934	0.560	0.146	0.231	0.147	0.192	0.265	0.306	0.416	0.426
25	0.934	0.560	0.158	0.242	0.151	0.194	0.266	0.310	0.416	0.426
20	0.934	0.560	0.159	0.243	0.150	0.194	0.268	0.311	0.416	0.426
15	**0.934**	**0.560**	**0.160**	**0.241**	**0.155**	**0.204**	**0.257**	**0.301**	**0.415**	**0.421**
10	0.934	0.560	0.141	0.221	0.142	0.178	0.255	0.302	0.414	0.387
8	0.934	0.560	0.151	0.254	0.138	0.172	0.242	0.291	0.393	0.386
6	0.945	0.550	0.141	0.223	0.146	0.178	0.233	0.271	0.376	0.365
4	0.904	0.509	0.137	0.182	0.152	0.145	0.223	0.241	0.356	0.383
2	0.803	0.522	0.111	0.097	0.125	0.057	0.247	0.210	0.360	0.193
VSM	**0.934**	**0.560**	**0.147**	**0.232**	**0.139**	**0.184**	**0.258**	**0.295**	**0.429**	**0.444**

4.3. Evaluation of Cluster-Based Retrieval Effectiveness

The experiments evaluate effectiveness of the TRSM cluster-based retrieval by comparing it with full retrieval by using the common measures of *precision* and *recall*. Precision, P, is the ratio of the number of relevant documents retrieved over the total number of documents retrieved. Recall, R, is the ratio of relevant documents retrieved for a given query over the number of relevant documents for that query in the database. Precision and recall are defined as

$$P = \frac{|\text{Rel} \cap \text{Ret}|}{|\text{Ret}|} \qquad R = \frac{|\text{Rel} \cap \text{Ret}|}{|\text{Rel}|} \tag{17}$$

where $\text{Rel} \subset \mathcal{D}$ is the set of relevant documents in the database for the query, and $\text{Ret} \subset \mathcal{D}$ is the set of retrieved documents. Table VI shows precision and recall of the TRSM-based full retrieval and the VSM-based full retrieval (vector space model[9]) where the TRSM-based retrieval is done with values 30, 25, 20, 15, 10, 8, 6, 4, and 2 of θ. After ranking all documents according to the query, precision and recall are evaluated on the set of retrieved documents determined by the default *cutoff* value as the average number of relevant documents for queries in each test collection. From this table we see that precision and recall for the JSAI are high, and they are higher and stable for the other collections with $\theta \geq 15$. With these values of θ, the TRSM-based retrieval effectiveness is comparable or somehow higher than that of VSM.

To evaluate the performance of cluster-based retrieval by the TRSM, we carried out retrieval experiments on all queries of test collections. For each query in the test collection, clusters are ranked according to the similarity between the query and the cluster representatives. Based on this ranking order, this the cluster-based retrieval is carried out.

Table VII reports the average of precision and recall for all queries in test collections using the TRSM cluster-based retrieval with 1, 2, 3, and 4 clusters, and full retrieval (20 clusters). Usually, along the ranking order of clusters, when cluster-based retrieval is carried out on more clusters, we obtain, higher recall value. Interestingly, the TRSM cluster-based retrieval achieved higher recall than that of full

Table VII. Precision and recall of the TRSM cluster-based retrieval.

	1 Cluster		2 Clusters		3 Clusters		4 Clusters		5 Clusters		Full search	
Col.	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*	*P*	*R*
JSAI	0.973	0.375	0.950	0.458	0.937	0.519	0.936	0.544	0.932	0.534	0.934	0.560
CACM	0.098	0.063	0.100	0.127	0.117	0.166	0.132	0.221	0.144	0.240	0.160	0.241
CISI	0.177	0.078	0.141	0.139	0.151	0.179	0.156	0.206	0.158	0.212	0.155	0.204
CRAN	0.204	0.219	0.238	0.278	0.250	0.290	0.257	0.301	0.261	0.304	0.257	0.301
MED	0.393	0.277	0.396	0.393	0.372	0.425	0.367	0.445	0.380	0.472	0.415	0.421

retrieval on several collections. More importantly, the TRSM cluster-based retrieval on four clusters offers precision higher than that of full retrieval in most collections. Also, the TRSM cluster-based retrieval achieved recall and precision nearly as high as that of full search just after searching on one or two clusters. These results show that the TRSM cluster-based retrieval can contribute considerably to the problem of improving retrieval effectiveness in information retrieval.

4.4. Evaluation of TRSM Nonhierarchical Clustering Efficiency

The proposed TRSM clustering algorithm in Table II has the linear time complexity $O(N)$ and space complexity $O(N)$, where N is the number of index terms in a text collection. The finding of the cluster representative C_k requires $O(|C_k|)$, therefore steps 1 and 3 are of complexity $O(M)$, where M is the number of documents in the collection. Step 2 is a linear pass with complexity $O(M)$. Step 4 repeats steps 2 and 3 in a limited number of iterations (in our experiments, step 4 terminated within 11 iterations of steps 2 and 3), and step 5 assigns unclassified documents once. Thus, the total time complexity of the algorithm is $O(N)$, because $M < N$.

However, the algorithm works on the base of data files associated with the TRSM described in Section 3. From a given collection of documents, we need to prepare all the files before running the TRSM nonhierarchical clustering algorithm. It consists of making an index term file, term encoding, document-term and term-document (inverted) relation files as indexing files, files of term co-occurrences, and tolerance classes for each value of θ. A direct implementation of these procedures requires the time complexity of $O(N^2)$, but we implemented the system by applying a sorting algorithm (quick-sort) of $O(N \log N)$ to make the indexing files, then created the TRSM-related files for the term co-occurrences, tolerance classes, upper, and lower approximations in the time of $O(N)$.

All the experiments reported in this article were performed on a conventional workstation GP7000S Model 45 (Fujitsu, 250 MHz Ultra SPARC-II, 512 MB). Theoretically, we can note that it requires on average m/K of the full search time, where K is the number of clusters.

Concerned with generating the TRSM files for the JSAI database, the direct implementation with $O(N^2)$ required up to 6 minutes (14 hours for CRAN), but the quick-sort-based implementation with $O(N \log N)$ took about 3 seconds (23 minutes for CRAN) for making the files by running a package of shell scripts on UNIX. The efficiency of the algorithm is shown in Table VII, where the TRSM time includes

Table VIII. Performance measurements of the TRSM cluster-based retrieval.

Col.	Size (MB)	No. of queries	TRSM time	Clustering time (sec)	Full search (sec)	1-Cluster search (sec)	Memory (MB)
JSAI	0.1	20	2.4 s	8.0	0.8	0.1	12
CACM	2.2	64	22 m 2.2 s	146.0	13.3	1.2	15
CISI	2.2	76	13 m 16.8 s	18.0	40.1	3.4	13
CRAN	1.6	225	23 m 9.9 s	13.0	20.5	1.8	13
MED	1.1	30	40.1 s	4.0	2.5	0.3	28

the time from processing the original texts until generating all necessary files input to the clustering algorithm. Thanks to a short time for preparing the database files, as well as a shorter time for a cluster-based search compared with the full search, the proposed method is able to be applied to large databases of documents.

5. CONCLUSION

We have proposed a document nonhierarchical clustering algorithm based on the tolerance rough set model of tolerance classes of index terms from document databases. The algorithm can be viewed as a kind of re-allocation clustering method where the similarity between documents is calculated using their tolerance upper approximations. Different experiments have been carried out on several test collections for evaluating and validating the proposed method on the clustering tendency and stability, the efficiency, and effectiveness of cluster-based retrieval using the clustering results. With the computational time and space requirements of $O(N\log N)$ and $O(N)$, the proposed algorithm is appropriate for clustering large document collections. The use of the tolerance rough set model and the upper approximations of documents allows us to use efficiently the method in the case when documents are represented by a few terms.

With the results obtained so far, we believe that the proposed algorithm contributes considerable features to document clustering and information retrieval. There is still much work to do in this research, such as (1) to incrementally update tolerance classes of terms and document clusters when new documents are added to the collections, and (2) to extend the tolerance rough set model by considering the model without requiring a symmetric similarity or tolerance classes based on co-occurrence between more than two terms.

Acknowledgments

The authors wish to thank anonymous reviewers for their valuable comments to improve this article.

References

1. Boyce BR, Meadow CT, Donald HK. Measurement in information science. Academic Press; 1994.
2. Croft WB. A model of cluster searching based on classification. Information System 1980:189–195.

3. Fakes WB, Baeza-Yates R, editors. Information retrieval. Data Structures and Algorithms. Prentice Hall; 1992.
4. Guivada VN, Raghavan VV, Grosky WI, Kasanagottu R. Information retrieval on the world wide web. IEEE Internet Computing 1997:58–68.
5. Ho TB, Funakoshi K. Information retrieval using rough sets. Journal of Japanese Society for Artificial Intelligence 1998;13(3):424–433.
6. Iwayama M, Tokunaga T. Hierarchical Bayesian clustering for automatic text classification. In: Proc 14th Joint Conference on Artificial Intelligence. Morgan Kaufmann Publishers; 1995. pp 1322–1327.
7. Lebart L, Salem A, Berry L. Exploring textual data. Kluwer Academic Publishers; 1998.
8. Lin TY, Cercone N, editors. Rough sets and data mining. Analysis of imprecise data. Kluwer Academic Publishers; 1997.
9. Manning CD, Schutze H. Foundations of statistical natural language processing. The MIT Press; 1999.
10. Pawlak Z. Rough sets: Theoretical aspects of reasoning about data. Kluwer Academic Publishers; 1991.
11. Polkowski L, Skowron A, editors. Rough sets in knowledge discovery 2. Applications, case studies and software systems. Physica-Verlag; 1998.
12. Raghavan VV, Sharma RS. A framework and a prototype for intelligent organization of information. Canadian Journal of Information Science 1986;11:88–101.
13. Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information Processing & Management 1988;4(5):513–523.
14. Skowron A, Stepaniuk J. Generalized approximation spaces. In: 3rd International Workshop on Rough Sets and Soft Computing. 1994. pp 156–163.
15. Slowinski R, Vanderpooten D. Similarity relation as a basis for rough approximations. In: Wang P, editor. Advances in machine intelligence and soft computing, 1997, Vol 4 pp 17–33.
16. Srinivasan P. The importance of rough approximations for information retrieval. International Journal of Man-Machine Studies 1991;34(5):657–671.
17. Van Rijsbergen CJ. A theoretical basis for the use of co-occurrence data in information retrieval. Journal of Documentation 1977;33(2):106–119.
18. Willet P. Similarity coefficients and weighting functions for automatic document classification: An empirical comparison. International Classification 1983;10(3):138–142.
19. Willet P. Recent trends in hierarchical document clustering: A critical review. Information Processing and Management 1988:577–597.