Cluster-based Information Retrieval with Tolerance Rough Set Model

Tu Bao Ho[1], Saori Kawasaki[1], and 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

Abstract

The objectives of this paper are twofold. First to introduce a model for representing documents with semantics relatedness using rough sets but with tolerance relations instead of equivalence relations (TRSM). Second to introduce two hierarchical and nonhierarchical document clustering algorithms based on this model and TRSM cluster-based information retrieval using these two algorithms. The experimental results show that TRSM offers an alternative approach to text clustering and information retrieval.

Keywords: tolerance rough set model, document clustering, information retrieval.

I. Introduction

Document clustering, the grouping of documents into several clusters, has been recognized as a means of improving efficiency and effectiveness of text retrieval. With the growing importance of electronic media for storing and exchanging large textual databases, document clustering becomes more significant. Document clustering helps the user to exploit large document collections in several ways, such as it enables the user to select and tackle only part of the collection that is relevant to his/her interest, or it assists the user from members and representatives of clusters to uncover topics, hypotheses, concepts, or novel nuggets. However, document clustering is a difficult clustering problem by a number of reasons [2], [4], [14]. The main difficulty comes from the unstructured form and textual characteristics of documents. As a consequence, the quality of document clustering not only depends on clustering algorithms but also largely depends on document representation models.

Rough set theory, a mathematical tool to deal with vagueness and uncertainty introduced by Pawlak in early 1980s [7], has been successful in many applications [5], [8]. 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 [9], [13]. 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 [3], ERSM is not suitable for information retrieval and text processing in general due to the fact that the requirement of the transitive property in equivalence relations is too strict to the meaning of words, and there is no way to calculate automatically equivalence classes of terms. Inspired by some works that employs different relations to generalize new models of rough set theory, e.g., [11], [12], a *tolerance rough set model* (TRSM) for text processing that adopts tolerance classes instead of equivalence classes has been developed [3].

In this work we extend TRSM in [3] and introduce two TRSM-based hierarchical and nonhierarchical document clustering algorithms, as well methods for cluster-based information retrieval (IR). These algorithms have been evaluated and validated experimentally on IR test collections. The results show advantages of the model, particularly in improving precision in information retrieval.

Section 2 of the paper presents the TRSM for representing documents. Section 3 describes two TRSM clustering algorithms. Section 4 presents a evaluation and validation of these algorithms, and section 5 addresses the TRSM cluster-based information retrieval.

II. The tolerance rough set model

Given a set $\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$ of M full text documents. Each document d_j is mapped into a list of terms t_i each is assigned a weight that reflects its im-

portance in the document. 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 are first calculated by

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

then normalized by $w_{ij} \leftarrow w_{ij} / \sqrt{\sum_{t_{hj} \in d_j} (w_{hj})^2}$. Each document d_j is represented by its r highest-weighted terms, i. e., $d_j = (t_{1j}, w_{1j}; t_{2j}, w_{2j}; \ldots; t_{rj}, w_{rj})$ where $w_{ij} \in [0, 1]$. A usual way is to fix a default value r common for all documents. The set of all terms from $\mathcal{D}$ and queries Q are denoted with $q_i \in \mathcal{T}$ and $w_{iq} \in [0, 1]$ by

$$\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$$
$$Q = (q_1, w_{1q}; q_2, w_{2q}; \ldots; q_s, w_{sq})$$

The *tolerance rough set model* (TRSM) aims to enrich the document representation in terms of semantics relatedness by creating tolerance classes of terms in $\mathcal{T}$ and approximations of subsets of documents. The model has the root from rough set models and its extensions [7], [11]. The key idea is among three properties of an equivalence relation R in an universe U used in the original rough set model (reflexive: xRx; symmetric: $xRy \rightarrow yRx$; transitive: $xRy \wedge yRz \rightarrow xRz$ for $\forall x, y, z \in U$), the transitive property does not always hold in natural language processing, information retrieval, and consequently text data mining. In fact, words are better viewed as overlapping classes which can be generated by *tolerance relations* (requiring only reflexive and symmetric properties).

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 an uncertainty function I depending on a threshold θ:

$$I_\theta(t_i) = \{t_j \mid f_{\mathcal{D}}(t_i, t_j) \geq \theta\} \cup \{t_i\} \quad (2)$$

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. A vague inclusion function ν, which determines how much X is included in Y, is defined as

$$\nu(X, Y) = \frac{|X \cap Y|}{|X|} \quad (3)$$

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

$$\mu(t_i, X) = \nu(I_\theta(t_i), X) = \frac{|I_\theta(t_i) \cap X|}{|I_\theta(t_i)|} \quad (4)$$

With these definitions we can define a tolerance space as $\mathcal{R} = (\mathcal{T}, I, \nu, P)$ 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

$$\mathcal{L}(\mathcal{R}, X) = \{t_i \in \mathcal{T} \mid \nu(I_\theta(t_i), X) = 1\} \quad (5)$$

$$\mathcal{U}(\mathcal{R}, X) = \{t_i \in \mathcal{T} \mid \nu(I_\theta(t_i), X) > 0\} \quad (6)$$

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.

The term-weighting method defined by Eq. (1) 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.

$$w_{ij} = \begin{cases} (1 + \log(f_{d_j}(t_i))) \times \log \frac{M}{f_{\mathcal{D}}(t_i)} & t_i \in d_j, \\ \min_{t_{hj} \in d_j} w_{hj} \times \frac{\log(M / f_{\mathcal{D}}(t_i))}{1 + \log(M / f_{\mathcal{D}}(t_i))} & t_i \in \mathcal{U}(\mathcal{R}, d_j) \setminus d_j \\ 0 & t_i \notin \mathcal{U}(\mathcal{R}, d_j) \end{cases} \quad (7)$$

Table 1: A document and its TRSM representation

MED_1: correlation between maternal and fetal plasma levels of glucose and free fatty acids . correlation coefficients have been determined between the levels of glucose and ffa in maternal and fetal plasma collected at delivery . significant correlations were obtained between the maternal and fetal glucose levels and the maternal and fetal ffa levels . from the size of the correlation coefficients and the slopes of regression lines it appears that the fetal plasma glucose level at delivery is very strongly dependent upon the maternal level whereas the fetal ffa level at delivery is only slightly dependent upon the maternal level .
MED_1: 21-0.178679, 44-0.094230, 48-0.228942, 57-0.235588, 110-0.257558, 198-0.328567, 299-0.126899, 403-0.371317, 437-0.136658, 683-0.306114, 692-0.306114, 694-0.306114, 1840-0.289422, 2546-0.189904, 4546-0.321535.

III. TRSM Clustering Algorithms

3.1 Algorithms

Table 2 describes the general TRSM-based hierarchical clustering algorithm that is an extension of the hierarchical agglomerative clustering algorithm. The main point here is at each merging step it uses upper

Table 2: TRSM hierarchical clustering algorithm

Input	$\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$ a similarity measure $\mathcal{S} : \mathcal{P}(\mathcal{D}) \times \mathcal{P}(\mathcal{D}) \rightarrow R^+$
Result	A hierarchical structure of $\mathcal{D}$

1. Initially, consider each document of $\mathcal{D}$ as a cluster with one member $C_j = \{d_j\}$, and $H = \{C_1, C_2, , \ldots, C_M\}$.
2. Identify two most similar clusters in terms of upper approximations of their representatives, $(C_{n_1}, C_{n_2}) = argmax_{(C_u, C_v) \in H \times H} \mathcal{S}(\mathcal{U}(\mathcal{R}, C_u), \mathcal{U}(\mathcal{R}, C_v))$
3. Form a new cluster $C_i = C_{n_1} \cup C_{n_2}$ and let $H = (H \setminus \{C_{n_1}, C_{n_2}\}) \cup \{C_i\}$.
4. If more than one cluster remains, return to steps 2 and 3.

Table 3: TRSM nonhierarchical clustering algorithm

Input	$\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$ and the number K.
Result	K clusters of documents with their membership.

1. Determine the initial representatives $R_1, R_2, ..., R_K$ of clusters $C_1, C_2, ..., C_K$ 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 $R_k, k = 1, ..., K$. 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, 4 and by NN(d_u) its nearest neighbor document (with non-zero similarity) in formed clusters. Assign d_u into the cluster that contains NN(d_u), and determine the cluster membership of d_u in this cluster as the product $m(d_u) = m(\text{NN}(d_u)) \times S(\mathcal{U}(\mathcal{R}, d_u), \mathcal{U}(\mathcal{R}, \text{NN}(d_u)))$. Re-determine the representatives R_k, for $k = 1, ..., K$.

approximations of documents in finding two closest clusters to merge. Several variants of agglomerative clustering can be applied, such single-link or complete-link clustering.

As documents are represented as length-normalized vectors and when cosine similarity measure is used, an efficient alternative is to employ the group-average agglomerative clustering. The group-average clustering avoids the elongated and straggling clusters produced by single-link clustering, and avoids the high cost of complete link clustering. In fact, it allows using cluster representatives to calculate the similarity between two clusters instead of averaging similarities of all document pairs each belong to one cluster [6], [14]. In such a case, the complexity of computing average similarity would be $O(N^2)$.

Table 3 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 [2].

The distinction of the TRSM nonhierarchical clustering algorithm is it forms overlapping clusters and it 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 reallocation 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.

We consider two other issues that have an important influence on the clustering quality (i) how to define the representatives of clusters; and (ii) how to determine the similarity between documents and the cluster representatives.

3.2 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:

(i) each document $d_j \in C_k$ has some or many terms in common with R_k;
(ii) terms in R_k are possessed by a large number of $d_j \in C_k$;
(iii) no term in R_k must be possessed by every document in C_k.

It is known that the Bayesian decision rule with minimum error rate will assign a document d_j in the cluster C_k if

$$P(d_j|C_k)P(C_k) > P(d_j|C_h)P(C_h), \forall h \neq k \qquad (8)$$

With the assumption that terms occur independently in documents, we have

$$P(d_j|C_k) = P(t_{j_1}|C_k)P(t_{j_2}|C_k) \ldots P(t_{j_p}|C_k) \qquad (9)$$

Denote by $f_{C_k}(t_i)$ the number of documents in C_k that contain t_i, we have $P(t_i|C_k) = f_{C_k}(t_i)/|C_k|$. Equation

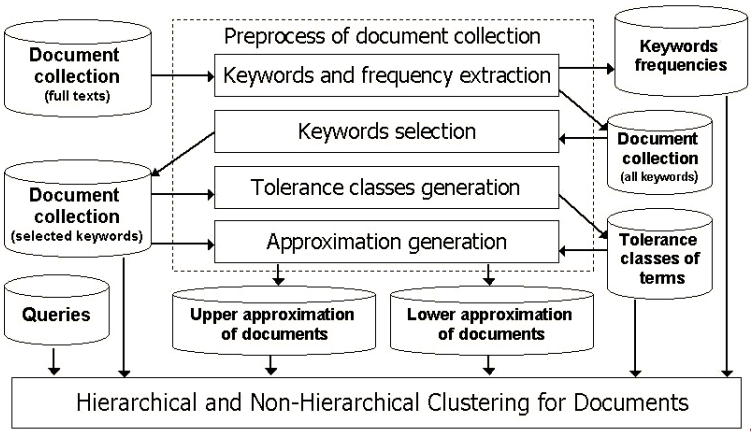

Figure 1: Conceptual architecture of the system

(9) and heuristics of the polythetic properties of the cluster representatives lead us to adopt rules to form the cluster representatives:

(i) Initially, $R_k = \phi$.
(ii) 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 $R_k = R_k \cup \{t_i\}$.
(iii) If $d_j \in C_k$ and $d_j \cap R_k = \phi$ then $R_k = R_k \cup \text{argmax}_{t_i \in d_j} w_{ij}$.

In case of group-average clustering, σ could be 0 to ensure the use of cluster representatives when calculating the cluster similarity. The weights of terms t_i in R_k is first averaged by of weights of this 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.

3.3 Document and cluster similarity

Many similarity measures between documents can be used in the TRSM clustering algorithm. Three common coefficients of Dice, Jaccard and Cosine [2] 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 cosine coefficient is

$$S_C(d_{j_1}, d_{j_2}) = \frac{\sum_{k=1}^{N}(w_{kj_1} \times w_{kj_2})}{\sqrt{\sum_{k=1}^{N} w_{kj_1} \times \sum_{k=1}^{N} w_{kj_2}}} \qquad (10)$$

It is worth to note that the cosine coefficient (or any other well-known similarity coefficient used for documents [2]) yields a large number of zero values when documents are represented by r terms as many of them may have no terms in common. 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 cosine coefficient is applied to the upper approximation of documents $\mathcal{U}(\mathcal{R}, d_j)$ and cluster representatives $\mathcal{U}(\mathcal{R}, R_k)$. Two main advantages of using upper approximations are: (i) To reduce the number of zero-valued coefficients by considering documents themselves together with the related terms in tolerance classes; and (ii) The upper approximations formed by tolerance classes make it possible to relate documents that may have few (even no) terms in common with the user's topic of interest or the query.

Table 4: Test collections

Collection	Subject	doc.	query	rel. doc.
JSAI	AI	802	20	32
CACM	Comp. Sci.	3200	64	15
CISI	Lib. Sci.	1460	76	40
CRAN	Aero.	1400	225	8
MED	Medicine	3078	30	23

IV. Validation and Evaluation

Table 4 summarizes test collections used in our experiments including JSAI where each document is represented in average by 5 keywords and four other common test collections CACM, CISI, CRAN and MED [2]. Columns 3, 4, and 5 show the number of documents, queries, and the average numbers of relevant documents for queries. The clustering quality for each test collection depends on parameter θ in TRSM and on σ in clustering algorithm. We can note that the higher value of θ the large upper approximation and the smaller lower approximation of a set X. Our experiments suggested that when the average number

Table 5: Results of clustering tendency

	% average of relevant documents						
	0	1	2	3	4	5	avg.
JSAI	19.9	19.8	18.5	18.5	11.8	11.5	2.2
CACM	50.3	22.5	12.8	7.9	4.2	2.3	1.0
CISI	45.4	25.8	15.0	7.5	4.3	1.9	1.1
CRAN	33.4	32.7	19.2	9.0	4.6	1.0	1.2
MED	10.4	18.7	18.6	21.6	19.6	11.1	2.5

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 9 we can see how retrieval effectiveness relates to different values of θ. To avoid biased experiments when comparing algorithms we take default values $\theta = 15$, and $\sigma = 0.1$ for all five test collections.

4.1 Validation of Clustering Tendency

The experiments for clustering tendency "attempt to determine whether worthwhile retrieval performance would be achieved by clustering a document collection, before investing the computational resources which clustering the database would entail" [2]. We employ the *nearest neighbor test* [14] 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 0, 1, 2, 3, 4, or 5 relevant documents in the set of 5 nearest neighbors of each relevant document. Table 5 reports the experimental results synthesized from those done on five test collections. Columns 4 and 5 show the number of queries and total number of relevant documents for all queries in each test collection. The next six rows stand for the percentage average of the relevant documents in a collection that had 0, 1, 2, 3, 4, and 5 relevant documents in their sets of 5 nearest neighbors. For example, the meaning of row JSAI column 11 is "among all relevant documents for 20 queries of JSAI collection, 11.5 % of them have 5 nearest neighbor documents are all relevant documents". The last column shows the average number of relevant documents among 5 nearest neighbors of each relevant document. This value is relatively high for JSAI and MED collections and vice-versa 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 if the TRSM clustering method might appropriately be applied for retrieval purpose. This tendency can be clearly observed in concordance with the high retrieval effectiveness for JSAI and MED shown in Table 9.

Table 6: Synthesized results about the stability

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

4.2 Validation of Clustering Stability

The experiments were done for the JSAI test collection in order to validate the stability of the TRSM clustering, i.e., to verify that whether the TRSM clustering method produces a hierarchy which is unlikely to be altered drastically when further documents are incorporated. For each value 2, 3, and 4 of θ, the experiments are done 10 times each for a reduced database of size $(100 - s)\%$ of $\mathcal{D}$. We randomly removed a specified amount of $s\%$ 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 6 synthesizes the experimental results with different values of s from 210 experiments with $s\%$ = 1%, 2%, 3%, 4%, 5%, 10% and 15%.

Note that a little change of data implies a possible little change of hierarchy (about at the same percentage as for $\theta = 4$). The experiments on the stability for other test collections have nearly the same results as those of JSAI. It suggests that the TRSM hierarchical clustering results are stable.

4.3 Hierarchical Clustering Efficiency

V. TRSM Cluster-based Information Retrieval

From a given collection of documents we need to prepare all the files before running the TRSM hierarchical clustering algorithm. It consists of making an index term file, term encoding, document-term and term-document (inverted) relation files as indexing files,

Table 7: Performance Measurements of the TRSM Cluster-based Retrieval

Col.	Size (MB)	Nb. of doc.	Nb. of query	TRSM Time	HC Time	NHC Time	Full Search	1-Cluster Search	HM (MB)	NHM (MB)
JSAI	0.1	802	20	2.4s	14.9s	8.0s	0.8s	0.1s	8	12
CACM	2.2	3200	64	22m2.2s	26m46.8s	2m26s	13.3s	1.2s	201	15
CISI	2.2	1460	76	13m16.8s	4m49.8s	18s	40.1s	3.4s	84	13
CRAN	1.6	1400	225	23m9.9s	3m6.9s	13s	20.5s	1.8s	71	13
MED	1.1	1033	430	0.1s	1m30.8s	4s	2.5s	0.3s	25	28

Table 8: Precision and recall of TRSM cluster-based retrieval and full retrieval

	1.2% (0.18)		1.8% (0.16)		2.9% (0.14)		8.0% (0.11)		16.9% (0.09)		full search	
Col.	P	R	P	R	P	R	P	R	P	R	P	R
JSAI	0.950	0.472	0.948	0.485	0.949	0.502	0.939	0.541	0.938	0.559	0.934	0.560
CACM	0.048	0.037	0.096	0.068	0.100	0.084	0.116	0.194	0.105	0.262	0.160	0.241
CISI	0.181	0.043	0.180	0.061	0.180	0.089	0.130	0.183	0.112	0.261	0.155	0.204
CRAN	0.121	0.127	0.140	0.149	0.139	0.173	0.139	0.214	0.112	0.245	0.257	0.301
MED	0.477	0.288	0.530	0.324	0.565	0.375	0.518	0.460	0.422	0.531	0.415	0.421
	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

files of term co-occurrences and tolerance classes for each value of θ. A direct implementation of these procedures requires the time complexity of $O(M + N^2)$, 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-occurrence, tolerance classes, upper and lower approximations files in the time of $O(M + N)$.

The experiments reported in this paper were performed on a conventional workstation GP7000S Model 45 (Fujitsu, 250 MHz Ultra SPARC-II, 512 MB). We can note that the search for clusters requires in average $\log M$, then the search will be done with a subset of documents in the clusters. However, the time complexity of the clustering is of $O(M^2+N)$, and the space is of $O(M^2 + N)$ because of using an $M \times M$-matrix to store the similarities/distances between clusters in the hierarchy. Concerned with generating the TRSM files for the JSAI database, the direct implementation with $O(M+N^2)$ required up to 6 minutes [14 hours for CRAN], but the quicksort-based implementation with $O(N \log N)$ took about 3 seconds (JSAI) [23 minutes for CRAN] for making the files by running a package of shell scripts on UNIX. Table 7 summaries the time for generating the TRSM files, clustering, full search, cluster-based search, and the required memory size for each collection. The clustering time included the time for reading the TRSM files into the RAM memory. Thanks to short time for preparing the database files as well as shorter time for cluster-based search in comparing with the full search, the TRSM-based proposed method is able to be applied to large document collections.

V. TRSM Cluster-based Information Retrieval

We show the potential of the method in terms of cluster-based effectiveness and efficiency in application to information retrieval [2], [14]. The quality of generated hierarchy is evaluated in terms of information retrieval. 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

$$P = \frac{|Rel \cap Ret|}{|Ret|} \qquad R = \frac{|Rel \cap Ret|}{|Rel|} \tag{11}$$

where $Rel \subset \mathcal{D}$ is the set of relevant documents in the database for the query, and $Ret \subset \mathcal{D}$ is the set of retrieved documents. Table 9 shows precision and recall of the TRSM-based full retrieval and the VSM-based full retrieval (Vector Space Model) where the

Table 9: Precision and recall of TRSM and VSM 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

Table 10: TRSM cluster-based retrieval vs. VSM cluster-based retrieval

	2.9% of $\mathcal{D}$ ($\gamma = 0.14$)				8.0% of $\mathcal{D}$ ($\gamma = 0.11$)				16.9% of $\mathcal{D}$ ($\gamma = 0.09$)			
	TRSM		VSM		TRSM		VSM		TRSM		VSM	
Col.	P	R	P	R	P	R	P	R	P	R	P	R
JSAI	0.949	0.502	0.947	0.501	0.939	0.541	0.947	0.518	0.938	0.559	0.939	0.549
CACM	0.100	0.084	0.075	0.479	0.116	0.194	0.075	0.479	0.105	0.262	0.075	0.479
CISI	0.180	0.089	0.099	0.366	0.130	0.183	0.099	0.366	0.112	0.261	0.099	0.366
CRAN	0.139	0.173	0.066	0.519	0.139	0.214	0.066	0.519	0.112	0.245	0.066	0.519
MED	0.565	0.375	0.520	0.430	0.518	0.460	0.458	0.521	0.422	0.531	0.375	0.585

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 Table 9 we see that precision and recall for 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.

5.1 Hierarchical cluster-based IR

We carried out retrieval experiments on all queries of test collections. Each query in the test collection is matched against the hierarchy from the root in the top-down direction in order to determine a subset $\mathcal{D}' \subset \mathcal{D}$. The subset $\mathcal{D}'$ is union of all clusters each has the similarity between the query and its representative greater than a threshold γ. The cluster-based retrieval is carried out in $\mathcal{D}'$. Table 9 reports the average of precision and recall for all queries in test collections using the TRSM cluster-based retrieval with various proportion (%) of $|\mathcal{D}'|$ to $|\mathcal{D}|$), and full retrieval in whole $\mathcal{D}$ (accordingly, values of γ). The results show that in several cases (JSAI, CISI, and MED) just searching a small part of $\mathcal{D}$, says 1.2% or 1.8%, TRSM cluster-based search reaches precision higher than that of full search. Also, the TRSM cluster-based search achieved recall higher than that of full retrieval on most collections when $|\mathcal{D}'|$ is about 17% of $|\mathcal{D}|$. Table 10 reports the effectiveness of TRSM cluster-based retrieval (TRSM) versus VSM cluster-based retrieval (VSM) when $|\mathcal{D}'|$ is 2.9%, 8.0%, and 16.9% of $|\mathcal{D}|$. It shows that TRSM cluster-based retrieval often achieves precision higher than that of VSM cluster-based retrieval thought its recall is somehow lower. The results suggest that TRSM can be used to improve precision of information retrieval, and so in a certain tasks of text mining.

5.2 Nonhierarchical cluster-based IR

The lower half of Table 8 reports the average of precision and recall for all queries in test collections using the TRSM cluster-based retrieval with 1, 2, 3, 4 clusters, and full retrieval (15 clusters). Usually, along the ranking order of clusters when cluster-based retrieval is carried out on the more clusters we obtain higher recall value. Interestingly, the TRSM cluster-based retrieval achieved higher recall than that of full 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 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.

VI. Conclusion

We have proposed a document hierarchical clustering algorithm based on the tolerance rough set model of tolerance classes of index terms. Careful experiments have been done on test collections for evaluating and validating the proposed method on the clustering tendency and stability, the efficiency as well as effectiveness of cluster-based retrieval using the clustering results. There are still many further works to do in this research: (1) to investigate the parameters of TRSM and their influence on text mining algorithms; (2) to incrementally update tolerance classes of terms and document clusters when new documents are added to the collections; (3) 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; and (4) to combine TRSM-based hierarchical and non-hierarchical clustering for very large text collections.

References

[1] Baeza-Yates, R. and Ribeiro-Neto, B., *Modern Information Retrieval*, Addison Wesley, 1999.

[2] Fakes, W. B. and Baeza-Yates, *Information Retrieval. Data Structures and Algorithms* (eds.), Prentice Hall, 1992.

[3] Ho, T. B. and Funakoshi K., "Information retrieval using rough sets", *Journal of Japanese Society for Artificial Intelligence*, Vol. 13, N. 3, pp. 424–433, 1998.

[4] Lebart, L., Salem, A., and Berry, L., *Exploring Textual Data*, Kluwer Academic Publishers, 1998.

[5] Lin, T. Y. and Cercone, N., *Rough Sets and Data Mining. Analysis of Imprecise Data* (eds.), Kluwer Academic Publishers, 1997.

[6] Manning, C. D. and Schutze, H., *Foundations of Statistical Natural Language Processing*, The MIT Press, 1999.

[7] Pawlak, Z., *Rough sets: Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, 1991.

[8] Polkowski, L. and Skowron, A., *Rough Sets in Knowledge Discovery 2. Applications, Case Studies and Software Systems* (eds.), Physica-Verlag, 1998.

[9] Raghavan, V. V. and Sharma, R.S., "A Framework and a Prototype for Intelligent Organization of Information", *The Canadian Journal of Information Science*, Vol. 11, pp. 88–101, 1986.

[10] Salton, G. and Buckley, C., "Term-Weighting approaches in automatic text retrieval", *Information Processing & Management*, Vol. 4, No. 5, pp. 513–523, 1998.

[11] Skowron, A. and Stepaniuk, J., "Generalized approximation spaces", *The 3rd International Workshop on Rough Sets and Soft Computing*, pp. 156–163, 1994.

[12] Slowinski, R. and Vanderpooten, D., "Similarity Relation as a Basis for Rough Approximations", *Advances in Machine Intelligence and Soft Computing*, P. Wang (ed.), Vol. 4, pp. 17–33, 1997.

[13] Srinivasan, P., "The importance of rough approximations for information retrieval", *International Journal of Man-Machine Studies*, Vol. 34, No. 5, pp. 657–671, 1991.

[14] Willet, P., "Recent trends in hierarchical document clustering: A critical review", *Information Processing and Management*, pp. 577–597, 1988.

Tu Bao Ho

Tu Bao Ho received a B.Eng. degree from Hanoi University of Technology in 1978, M.S. and Ph.D. degrees from University Paris 6, in 1984 and 1987, and Habilitation from University Paris 9 in 1998. He has been with JAIST as a visiting associate professor at School of Information Science since July 1993, and as a professor at School of Knowledge Science since April 1998. His research interests include machine learning, knowledge-based systems, knowledge discovery and data mining.

Phone & Fax: +81-761-51-1730
Email: bao@jaist.ac.jp

Saori Kawasaki

Email: skawasa@jaist.ac.jp

Ngoc Binh Nguyen

Email: binh@jaist.ac.jp