修　士　論　文

A New Hierarchical Clustering Algorithm for Documents based on Tolerance Rough Set Model

指導教官　　Ho Tu Bao　教授

北陸先端科学技術大学院大学
知識科学研究科知識システム基礎学専攻

950026　河崎　さおり

審査委員：　Ho Tu Bao　教授　（主査）
石崎　雅人　助教授
中森　義輝　教授

2000 年 8 月

A New Hierarchical Clustering Algorithm for documents based on Tolerance Rough Set Model

By Saori Kawasaki

A thesis submitted to
School of Knowledge Science,
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Master of Knowledge Science
Graduate Program in Knowledge Science

Written under the direction of
Professor Tu Bao Ho

August 15, 2000

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Document Clustering

Text clustering, the grouping of texts into several clusters, has been used as a means of improving both the efficiency and the effectiveness of information retrieval and text processing [Lebart 98], [Larsen 99], [Baeza-Yates 99], [Manning 99]. About efficiency, it is clear that it helps to reduce the number of comparisons for searching similar documents in appropriate clusters when the clusters are evaluated closed enough to the requests. About effectiveness, it is also assumed that clustering can improve the accuracy of retrieval, but this assumption has not been verified yet [Iwayama 95].

There are two main types of clustering methods. One is *non-hierarchical clustering*, which often divides a data set of *M* items into *K* disjoint clusters and the other is *hierarchical clustering*, which produces a nested data set in which pairs of items or clusters are successively linked until every item in the data set is connected [Fakes 92], [Manning 99].

Hierarchical clustering methods do not require *a priori* knowledge of the number of clusters or the starting partition that are required in case of non-hierarchical clustering methods. Clusters in a hierarchy can be somehow more flexible to determine the target spaces than clusters divided in a non-hierarchical spaces, while non-hierarchical clustering requires much smaller computational costs than the hierarchical clustering [Sharma 96]. Because of the usefulness, hierarchical clustering has become more dominant in document

clustering. However, document clustering is a difficult clustering problem by a number of reasons [Fakes 92], [Lebart 98], [Willet 88], and some problems occur additionally when doing clustering on large textual databases. Particularly, when documents in a large textual database are represented by only a few keywords or they have few index terms in common, current available similarity measures in textual clustering [Boyce 94], [Fakes 92] often yield zero-values that decreases considerably the clustering quality.

1.2 Tolerance Rough Set Model and Clustering

In contrast to the *exact match* techniques for information retrieval, there have been many works on *inexact match* (partial) techniques that employ semantic calculations in order to improve the effectiveness of information retrieval. Fuzzy clustering [Miyamoto 90], latent semantic indexing [Manning 99] are among inexact techniques for information retrieval. Many inexact match methods employ expressions by co-occurrence of terms, categories dictionary and phrase pattern matching because of their significances [石岡 98], [那須川 99], [笠井 99], [関根 99].

Rough set theory, a mathematical tool to deal with vagueness and uncertainty introduced by Pawlak in early 1980s [Pawlak 91], has been successful in many applications [Lin 97], [Polkowski 98]. 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 *equivalence rough set model* (ERSM), has been investigated by several researchers [Raghavan 86], [Srinivasan 91]. 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 [Ho 98], ERSM is not suitable for information retrieval and text

processing 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., [Skowron 94], a *tolerance rough set model* (TRSM) for text processing that adopts tolerance classes instead of equivalence classes has been developed [Ho 98].

This model makes tolerance spaces of terms and documents according to term co-occurrences in each document and in the whole database. The most important contribution of TRSM is that it can calculate the semantic relations between documents using upper approximations even if documents do not share common terms. However, the original work on TRSM do not address the problem of term weighting that has an important role in enriching the document representation in text processing [Fakes 92], [Baeza-Yates 99]. An extension of TRSM with term weighting techniques [Salton 88], [藤田 99] could improve considerably the applicability of this model in text processing.

This research concerns with the question of whether TRSM can be applied to text clustering in order to achieve a better effectiveness and efficiency in information retrieval, and possibly to other tasks of text processing. Based on the general hierarchical agglomerative clustering scheme, a TRSM-based algorithm is developed. During the process of constructing a hierarchy, representatives with weighted terms are used to improve the computing cost of similarity between clusters instead of calculation of all document pairs.

The algorithm has been implemented, evaluated and validated on a number of common test collections. Experimental results show that the TRSM-based algorithm can be a promising alternative to do the task of hierarchical clustering for documents.

1.3 Organization of the Thesis

This paper reports the work on the TRSM-based hierarchical clustering algorithm for documents. Chapter 2 gives a brief recall of the basic notions of document clustering and the tolerance rough set model for documents. Chapter 3 describes details of the algorithm consisting of two phases of pre-processing for documents and hierarchical clustering. Chapter 4 is concerning with the implementation of the algorithm. Chapter 5 reports the evaluation and validation of the algorithm on a number of common test collections. Chapter 6 gives a conclusion of the research and future work. The appendix provides source code of the programs.

Chapter 2
Rough Sets and
Tolerance Rough Set Model (TRSM)

This chapter recalls basic notions of rough sets and tolerance rough set model (TRSM).

2.1 Basic Notions of Rough Sets

The rough set theory was introduced by Pawlak in early 1980s as a mathematical tool to deal with vagueness and uncertainty [Pawlak 91]. It has been applied in many fields such as machine learning, knowledge acquisition, decision analysis, information retrieval, pattern recognition, and recently knowledge discovery and data mining [中村 96], [Lin 97], [Polkowski 98].

2.1.1 Approximation by Equivalence Spaces

The starting point of theory of rough sets is that each set X in a universe U of objects can be "viewed" approximately by its upper and lower approximations in an *equivalence space* $R = (U, R)$ 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 R of any subset X of U, denoted respectively by $L(R, X)$ and $U(R, X)$, are defined by

$$L(R, X) = \{x \in U: [x]_R \subseteq X\}$$
$$U(R, X) = \{x \in U: [x]_R \neq \varnothing\}$$

where $[x]_R$ denotes the equivalence class of objects indiscernible with x regarding the equivalence relation R. This original model of rough sets is called the *equivalence rough set model* (ERSM) as it employs equivalent relations as its basics.

The following example illustrates the basic notions of rough sets. Table 2.1 shows a small universe of eight cells,

$$U = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$$

each of them belongs to one of two classes “healthy” and “cancerous”. The cells are described by three attributes: attribute “color” with two possible values “dark” and “light”; attribute “number of nuclei” with two values “1” and “2”; and attribute “number of tails” also with two values “1” and “2”.

Table 2-1: Examples of healthy and cancerous cells

	color	number of nuclei	number of tails	Class
C_1	light	1	1	healthy
C_2	dark	1	1	healthy
C_3	light	1	2	healthy
C_4	light	2	1	healthy
C_5	dark	1	2	cancerous
C_6	dark	2	1	cancerous
C_7	light	2	2	cancerous
C_8	dark	2	2	cancerous

Consider an equivalence relation R determined by two attributes “color” and “number of nuclei”. The approximation space regarding this equivalence relation R consists of four equivalence classes,

$$R = (U, R) = \{\{C_1, C_3\}, \{C_2, C_5\}, \{C_4, C_7\}, \{C_6, C_8\}\}.$$

Here, for example, C_2 and C_3 are indiscernible regarding the relation R (in terms of two attributes "color" and "number of nuclei"), and if consider the set $X = \{C_1, C_2, C_3\}$, we have its approximations

$$L(R, X) = \{C_1, C_3\}$$
$$U(R, X) = \{C_1, C_2, C_3, C_5\}$$

2.1.2 Applications of Rough Sets to Text Processing

The traditional rough set theory with equivalence relations concerns disjoint classes of objects because the equivalence relations require three properties that are reflexive, symmetric and transitive. A relation $R \subseteq U \times U$ is equivalent if it satisfies the following properties for all $x, y, z \in U$

(i) Reflexive: xRx
(ii) Symmetric: $xRy \rightarrow yRx$
(iii) Transitive: $xRy \wedge yRz \rightarrow xRz$

Rough sets have been applied to information retrieval and text processing since its early days. All early work on information retrieval using rough sets was based on ERSM with a basic assumption that the set T of index terms can be divided into equivalence classes determined by equivalence relations [Raghavan 86], [Srinivasan 89], [Srinivasan 91].

These properties are often hold in many application fields but all the properties do not always hold in certain application domains. Especially in the fields of linguistic or information retrieval, the transitive property is too strict and rough sets using the equivalence relation cannot be well applied. 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 2-1 shows part of associated classes of three words *root*, *cause* and *basis*. It is clear that these classes are not disjoint (equivalence classes) but *overlapping* as the meaning of the words is not transitive.

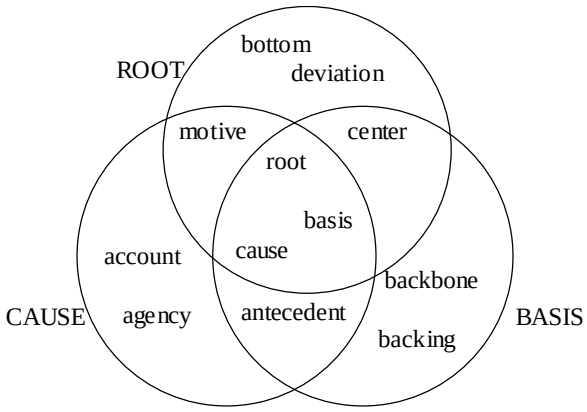


Figure 2-1: Overlapping classes of words

Intelligent information retrieval methods rely mainly on the conceptual analysis and they require considering semantics relationships between terms. Sometimes each term can be replaced by another term with similar concept, but cannot be always replaced by the same term because there is no such thing as a *true* synonym that have exactly the same meaning. Only case that the term can be always replaced by another term that is indiscernible to the original term is the synonyms such as a relation between "artificial intelligence" and "AI".

2.2 Generalized Tolerance Spaces

2.2.1 Tolerance Relations and Tolerance Spaces

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 [Skowron 94] in which generalized spaces are called *tolerance spaces* that contain overlapping classes of objects in the universe called *tolerance classes*. A tolerance space is generally defined as a quadruple $R = (U, I, \nu, P)$, 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.

Assume that an object x is perceived by information $Inf(x)$ about it. The certainty function $I: U \rightarrow 2^U$ determines $I(x)$ as a tolerance class of all objects which are considered to have *similar* information with x. This uncertainty function can be any function satisfying the condition $x \in I(x)$ and $y \in I(x)$ if and only if $x \in I(y)$ for any elements $x, y \in U$. Such a function corresponds to a relation $I \subseteq U \times U$ understood as xIy if and only if $y \in I(x)$. I is a tolerance relation because it satisfies the properties of reflexivity and symmetry.

The vague inclusion $\nu: 2^U \times 2^U \rightarrow [0,1]$ 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, i.e., $\nu(X, Y) \leq \nu(X, Z)$ for any $X, Y, Z \subseteq U$ and $Y \subseteq Z$.

Finally, the structurality function is introduced by analogy with mathematical morphology [Skowron 94]. In the construction of the lower and upper approximations, only tolerance sets being structural elements are considered. We define that $P: I(U) \rightarrow \{0,1\}$ 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 $L(R, X)$ and the upper approximation $U(R, X)$ in R of any $X \subseteq U$ are defined as

$$L(R, X) = \{x \in U: P(I(x)) = 1 \ \& \ \nu(I(x), X) = 1\}$$
$$U(R, X) = \{x \in U: P(I(x)) = 1 \ \& \ \nu(I(x), X) > 0\}$$

2.2.2 Example of Tolerance Space

Consider the examples of eight healthy and cancerous cells shown in Table 2-1. A tolerance space $R = (U, I, \nu, P)$ can be given as follows.

- $U = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$
- The uncertainty function $I: U \rightarrow 2^U$ is defined as

  $I(C_1) = \{C_1, C_2\}$
  $I(C_2) = \{C_2, C_1\}$
  $I(C_3) = \{C_3, C_4, C_5\}$
  $I(C_4) = \{C_4, C_3, C_7\}$
  $I(C_5) = \{C_5, C_3, C_8\}$
  $I(C_6) = \{C_6, C_8\}$
  $I(C_7) = \{C_7, C_4\}$
  $I(C_8) = \{C_8, C_5, C_6\}$

- The vague inclusion $\nu: 2^U \times 2^U \rightarrow [0,1]$ is defined as

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

- Define $P(I(C_k)) = 1$ for all $C_k \in U$.

It is easy to verify that the functions I, ν, and P satisfy all required properties, and they form a tolerance space R with the following tolerance classes

$R = \{\{C_1, C_2\}, \{C_3, C_4, C_5\}, \{C_4, C_3, C_7\}, \{C_5, C_3, C_8\}, \{C_6, C_8\}, \{C_7, C_4\}, \{C_8, C_5, C_6\}\}$

It is easy to verify that these classes satisfy two properties of symmetric and reflexive. In this tolerance space, if the set $X = \{C_1, C_2, C_3\}$, we have

$$L(R, X) = \{C_1, C_2\}$$
$$U(R, X) = \{C_1, C_2, C_3, C_4, C_5, C_7, C_8\}$$

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

2.3 A Tolerance Rough Set Model for Documents

2.3.1 Determination of a Tolerance Rough Set Model

In [Ho 98], the authors proposed a *tolerance rough set model* (TRSM) for documents. Essentially, they showed how to determine suitably I, ν and P for universes of documents in problems such as information retrieval, text data mining, etc.

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

First of all, to define a tolerance space R , the universe U is chosen as the set T of all index terms

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

The most crucial issue in formulating a TRSM for documents is identification of tolerance

classes of index terms. There are several ways to identify conceptually similar index terms, e.g., human experts, thesaurus, term co-occurrence, etc. The co-occurrence of index terms in all documents from D to determine a tolerance relation and tolerance classes is employed. The co-occurrence of index terms is chosen for the following reasons (see also [Van Rijsbergen 77]):

(i) It gives a meaningful interpretation in the context of information retrieval about the dependency and the semantic relation of index terms;

(ii) 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 $f_D(t_i, t_j)$ the number of documents in D in which two index terms t_i and t_j co-occur.

The uncertainty function I depending on a threshold θ is defined as

$$I_\theta(t_i) = \{t_j \mid f_D(t_i, t_j) \geq \theta\} \cup \{t_i\}$$

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

The vague inclusion function ν is defined as

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

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

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

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

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

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

2.3.2 Illustration of Tolerance Rough Set Model

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).

Table 2-2: Approximations of first 10 documents concerning “machine learning”

	Index terms	$L(R, X)$	$U(R, X)$
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_4, t_{15}	t_4, t_{15}	$t_1, t_2, t_4, 5, t_{15}, t_{26}$
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}	$t_1, t_{16}, t_{21}, t_{26}, t_{29}, t_{30}, t_{31}$	$t_{16}, t_{21}, t_{26}, t_{29}, t_{30}, t_{31}$	$t_1, t_2, t_5, t_{16}, t_{21}, t_{26}, t_{29}, t_{30}, t_{31}$

The JSAI database consists of 802 documents. In total, there are 1823 keywords in the database, and each document has in average 5 keywords. To illustrate the introduced notions, let us consider a part of this database that consists of first 10 documents concerning “machine learning” (two first columns of Table 2-2)

The keywords in this small universe are indexed by their order of appearance, i.e.,

t_1 = “machine learning”,
t_2 = “knowledge acquisition”
…
t_{30} = “neural networks”
t_{31} = “logic programming”

With $\theta = 2$ by definition of tolerance class we have tolerance classes of index terms

$I_2(t_1) = \{t_1, t_2, t_5, t_{16}\}$
$I_2(t_2) = \{t_1, t_2, t_4, t_5, t_{26}\}$
$I_2(t_4) = \{t_2, t_4\}$
$I_2(t_5) = \{t_1, t_2, t_5\}$
$I_2(t_6) = \{t_6, t_7\}$
$I_2(t_7) = \{t_6, t_7\}$
$I_2(t_{16}) = \{t_1, t_{16}\}$
$I_2(t_{26}) = \{t_1, t_{26}\}$

and each of other index terms has the corresponding tolerance class consisting of only itself, e.g., $I_2(t_3) = \{t_3\}$. Table 2-2 shows these 10 documents, their lower and upper approximations with $\theta = 2$.

Chapter 3
The TRSM-based Hierarchical Clustering Algorithm

Given a set *D* of *M* full text documents. The TRSM-based hierarchical clustering algorithm for generating a hierarchical structure of *D* consists of two phases. The first phase represents full documents by its surrogates in terms of TRSM (extracts and maps each document into a set of terms, then enriches documents with their approximations by the proposed tolerance rough set model). The second phase clusters documents by an agglomerative clustering method using the document approximations.

3.1 Representing Documents by TRSM

3.1.1 TRSM weighted representation of documents

The tolerance rough set model aims to enrich the document representation in terms of semantics relatedness by creating tolerance classes of terms in *T* and approximations of documents or subsets of documents. In the first phase each document d_j is mapped into a list of terms t_i each is assigned a weight that reflects its importance in the document. Several techniques have been used to filter irrelevant terms (such as particles “an”, “the”, etc.) and to identify terms that have the same stem (such as “driving” and “drive”). A document is a sequence of words, and there are several ways to represent it in the smaller

size than the original like text summarization or the indexing by key terms. The *vector space model* (VSM) that represents a document by a vector of terms with their weights determined according to their importance in the document is commonly used [Manning 99]. The following notations are used: $f_{dj}(t_i)$ denotes the number of occurrences of term t_i in d_j (term frequency), $f_D(t_i)$ denotes the number of documents in D that term t_i occurs in (document frequency). The weights w_{ij} of terms t_i in document d_j are first calculated by

$$w_{ij} = \begin{cases} (1+\log(f_{d_j}(t_i)))\times\log\dfrac{M}{f_D(t_i)} & if\ t_i \in d_j \\ 0 & if\ t_i \notin U(R, d_j) \end{cases}$$

as adopted from [Salton 88], [Baeza-Yates 99], [Manning 99]. The formula $\log M/f_D(t_i)) = \log M - \log f_D(t_i)$ gives full weight to words that occur in 1 document ($\log M - \log f_D(t_i) = \log M - \log 1 = \log M$). A word that occur in all documents would get zero weight ($\log M - \log f_D(t_i) = \log M - \log M = 0$). These weights are then normalized by vector length as

$$w_{ij} \leftarrow \frac{w_{ij}}{\sqrt{\sum_{t_h \in d_j} (w_{hj})^2}}$$

In fact, each document d_j is represented by its r highest-weighted terms. A usual way is to fix a default value r commonly for all documents (r = 15 in our experiments).

The term-weighting method is extended in the context of TRSM in order to define weights for terms in the upper approximation $U(R, d_j)$ 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.

$$w_{ij} = \begin{cases} (1+\log(f_{d_j}(t_i)))\times\log\dfrac{M}{f_D(t_i)} & \text{if } t_i \in d_j \\ \min\limits_{t_h \in d_j} w_{hj} \times \dfrac{\log(M / f_D(t_i))}{1+\log(M / f_D(t_i))} & \text{if } t_i \in U(R, d_j) \setminus d_j \\ 0 & \text{if } t_i \notin U(R, d_j) \end{cases}$$

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

3.1.2 Illustration of Document Representation

The representation of documents is illustrated here by using the well-known test collection MED in the field of medicine [Fox 90]. The collection consists of 1033 documents where first two documents are shown in Table 3-1.

Table 3-1: The first two documents in MED

I 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 . I 3 surfactant in fetal lamb tracheal fluid . lambs delivered by cesarean section with intact fetal circulation have a fluid filling the trachea . analysis revealed that this fluid contained material high in surface activity in lambs delivered near term, but less surface activity in premature lambs . administration of 10 per cent oxygen to the ewe for 1 hour prior to delivery did not alter the surfactant properties of the fetal tracheal fluid . two analyses of the fetal tracheal fluid revealed it to contain 146 and 198 mg. of lipid per 100 ml., 30 to 40 per cent of which was phospholipid, part of the active component of surfactant . the investigations reported here offer a model for further research into possible intrauterine factors in the pathogenesis of hyaline membrane disease .

Table 3-2: Tolerance classes of first 10 terms in MED with highest frequencies ($\theta = 10$)

1:	1,2,3,5,6,8,11,12,25,27,28,34,35,38,49,52,75,88,104,112,
2:	1,2,4,8,13,16,21,49,
3:	1,3,8,10,11,25,64,
4:	2,4,6,14,20,30,32,51,61,144,
5:	1,5,
6:	1,4,6,
7:	7,22,37,
8:	1,2,3,8,13,23,24,91,
9:	9,
10:	3,10,59,

Table 3-3: First 4 documents in MED represented by 15 pairs of term-weight (in order of terms) and 15 pairs of term#frequency (in order of decreasing frequency)

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,
MED_2 :	37-0.132605, 54-0.174948, 58-0.200884, 125-0.195653, 130-0.198301, 131-0.225484, 179-0.275673, 249-0.176005, 253-0.353217, 313-0.264077, 1660-0.246788, 1776-0.347819, 1992-0.274171, 3005-0.274171, 3069-0.386412,
MED_3 :	96-0.195176, 159-0.124811, 170-0.214596, 234-0.133426, 251-0.348168, 307-0.235155, 343-0.142042, 403-0.347371, 457-0.149568, 736-0.349057, 841-0.349057, 1054-0.174943, 1674-0.187086, 2229-0.436183, 2230-0.207844,
MED_4 :	9-0.158404, 46-0.139281, 200-0.170859, 695-0.298450, 725-0.220623, 791-0.220623, 1068-0.232051, 1100-0.287621, 1301-0.307585, 1401-0.248158, 1823-0.248158, 2461-0.341713, 3250-0.275692, 3675-0.341713, 4385-0.275692,
MED_1:	198#6,403#6,57#4,48#4,110#4,21#3,683#3,692#3,694#3,1840#2,4546#2, 2546#1,437#1,299#1,44#1,
MED_2:	253#11,179#6,131#4,313#4,3069#4,1776#4,58#4,130#3,54#3,125#3,1992#2, 37#2,1660#2,3005#2,249#2,
MED_3:	251#5,403#4,2229#3,736#3,841#3,170#2,307#2,96#2,159#1,1674#1,343#1, 457#1,234#1,1054#1,2230#1,
MED_4:	2461#3,3675#3,1301#3,1100#3,695#4,9#4,791#2,1401#2,1068#2,46#2,3250#2, 725#2,200#2,4385#2,1823#2,

Table 3-4: Upper (first half) and lower (second half) approximations of the first 4 documents in MED with $\theta = 10$.

MED_1:	4546,2546,1840,694,692,683,437,403,299,222,198,110,81,57,48,44,24,21,9,2,
MED_2:	3069,3005,1992,1776,1660,418,313,294,253,249,179,131,130,129,125,100,58, 54,37,22,19,17,14,
MED_3:	2230,2229,1674,1054,841,736,457,403,343,307,251,234,170,159,96,
MED_4:	4385,3675,3250,2461,1823,1401,1301,1100,1068,791,725,695,200,182, 134, 86, 82,73,61,51,46,41,21,18,9,4,
MED_1:	198,403,48,683,692,694,1840,4546,2546,437,299,
MED_2:	253,179,313,3069,1776,130,54,125,1992,1660,3005,
MED_3:	251,403,2229,736,841,170,307,159,1674,343,457,234,1054,2230,
MED_4:	2461,3675,1301,1100,695,791,1401,1068,3250,725,200,4385,1823,

Table 3-2 shows the tolerance classes with $\theta = 10$ for first 10 terms. It is easy to verify partially the reflexive and symmetric of the tolerance classes for these first 10 terms by their elements belonging to first 10 terms. Table 3-3 illustrates first 4 documents represented by 15 pairs of term-weight and 15 pairs of term#frequency, and Table 3-4 illustrates the lower and upper approximations of first 4 documents with $\theta = 10$.

3.2 A TRSM-based Hierarchical Agglomerative Clustering Method (HACM)

3.2.1 The TRSM-based HACM Algorithm

There are two approaches for creating a hierarchy: agglomerative (bottom-up) or divisive (top-down). The former begins from an un-clustered data set with *N*-1 pair-wise joins. The

latter begins with all objects in a single cluster and progresses through N-1 divisions of some cluster into a smaller cluster. The divisive methods are less commonly used and few algorithms are available, particularly in document clustering because it is difficult to decide how to divide the nodes that are close to the root. Therefore, the agglomerative method is employed for the TRSM approach for documents.

Figure 3-1 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 where upper approximations of documents are used in finding two closest clusters to merge by group-average link, which allows to use cluster representatives to calculate the similarity between clusters instead of averaging similarities of all document pairs included in clusters with average complexity $O(N^2)$. In such a case, the complexity of computing average similarity would be $O(N^2)$.

Input : a collection of M documents $D = \{d_1, d_2, ..., d_M\}$

a similarity measure $S : 2^D \times 2^D \rightarrow \Re$

Output : Hierarchical structure H of subsets C_k of D

1. Initially, consider each document d_j in D as a cluster with one member $C_j = \{d_j\}$ and $H = \{C_1, C_2, ..., C_M\}$.
2. Identify two most similar clusters in terms of upper approximations of their representatives, that means $(C_{n1}, C_{n2}) = \underset{(C_u, C_v) \in H \times H}{\arg\max} S(U(R, R_u), U(R, R_v))$
3. Form a new cluster $C_k = C_{n1} \cup C_{n2}$ and let $H = (H \setminus \{C_{n1}, C_{n2}\}) \cup \{C_k\}$
4. If more than one cluster remains, return to steps 2 and 3.

Figure 3-1: The general TRSM-based hierarchical agglomerative clustering algorithm

Two other issues, which have an important influence on the clustering quality, are to be considered:

(i) How to define the representatives of clusters; and

(ii) How to determine the similarity between documents and the similarity between clusters.

3.2.2 Representatives of Clusters

The representative of each cluster is a set of terms that characterizes documents of this cluster. Representatives of clusters play an important role in improving the usage of clustering results. They allow us to distinguish quickly clusters as well to calculate the distance between them. As the representatives of clusters are themselves sets of terms, they can be represented as documents that means by terms and their weights.

The TRSM hierarchical 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

$$P(d_j \mid C_k)P(C_k) > P(d_j \mid C_h)P(C_h), \forall h \neq k$$

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

$$P(d_j \mid C_k) = P(t_{j1} \mid C_k)P(t_{j2} \mid C_k)...P(t_{jp} \mid C_k)$$

Denote by $f_{Ck}(t_i)$ the number of documents in C_k that contain t_i, we have the probability $P(t_i|C_k) = f_{Ck}(t_i)/| C_k |$. The last equation and heuristics of the polythetic properties of the cluster representatives lead us to adopt rules to form the cluster representatives:

(i) Initially, $R_k = \varnothing$,

(ii) For all $d_j \in C_k$ and for all $t_i \in d_j$, if $f_{Ck}(t_i) / |C_k| > \sigma$ then $R_k = R_k \cup t_i$,

(iii) If $d_j \in R_k$ and $d_j \cap R_k = \varnothing$ then $R_k = R_k \cup \text{argmax}_{dj \in Ck} w_{ij}$

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 these terms in all documents belonging to C_k, that means

$$w_{ik} = (\sum_{d_j \in C_k} w_{ij}) / |\{d_j : t_i \in d_j\}|$$

These weights are then normalized by length of the representative R_k.

3.2.3 Distances between Documents

The distance is one of most important issues in clustering. In case items have numeric values, it is easy to calculate the distance between two items. But text cannot be calculated like numeric items. Several coefficients are used as substitutions. Famous and common coefficients are Dice, Jaccard and Cosine [Fakes, 92], [Boyce 94]. All of them employ the number of terms commonly appear in both documents. Following are definitions of those coefficients between a pair of documents d_{j1} and d_{j2}.

Dice coefficient:

$$S_D(d_{j1}, d_{j2}) = \frac{2 \times \sum_{k=1}^{N} (w_{kj1} \times w_{kj2})}{\sum_{k=1}^{N} w_{kj1}^2 + \sum_{k=1}^{N} w_{kj2}^2}$$

Jaccard coefficient:

$$S_J(d_{j1}, d_{j2}) = \frac{\sum_{k=1}^{N} (w_{kj1} \times w_{kj2})}{\sum_{k=1}^{N} w_{kj1}^2 + \sum_{k=1}^{N} w_{kj2}^2 - \sum_{k=1}^{N} (w_{kj1} \times w_{kj2})}$$

Cosine coefficient:

$$S_C(d_{j1}, d_{j2}) = \frac{\sum_{k=1}^{N}(w_{kj1} \times w_{kj2})}{\sqrt{\sum_{k=1}^{N} w_{kj1}^2 \times \sum_{k=1}^{2} w_{kj2}^2}}$$

The *Dice coefficient* normalizes for length by dividing by the total number of non-zero entries and multiplying by 2 can give a measure ranges from 0.0 to 1.0 with 1.0 indicating identical vectors. The *Jaccard coefficient* penalizes a small number of shared entries (as a proportion of all non-zero entries) more than the *Dice coefficient* does. The efficient becomes lower in low-overlap cases. The *cosine coefficient* is identical to the *Dice coefficient* for vectors with the same number of non-zero entries, but it penalizes less in cases where the number of non-zero entries is very different. Those coefficient values show how much two documents are similar and it is clear that zero similarity occurs when there is no common word in both documents even if they have terms of similar meaning. In this algorithm, approximations, especially upper approximations, of are used for the distance calculation for avoiding zero similarity documents instead of original terms included in documents. Further, the weights are employed for describing those coefficients in order to reflect the importance of terms more precisely.

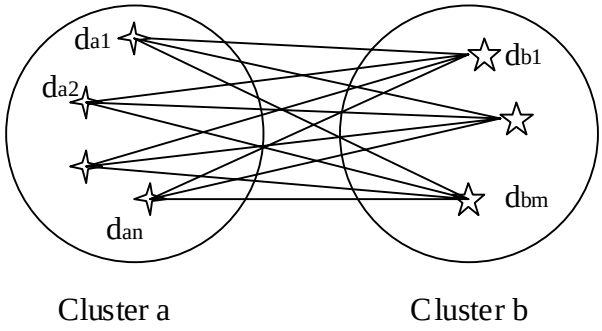
We adopt the similarity $S(d_{j1}, d_{j2})$ between two documents d_{j1} and d_{j2} that is defined by cosine coefficient, and the distance $D(d_{j1}, d_{j2})$ is defined as :

$$D(d_{j1}, d_{j2}) = 1 - S(d_{j1}, d_{j2})$$

3.2.4 Distances between Clusters

There are several ways to measure the distance between clusters. The most commonly used measures are single-link, complete-link, group-average-link and Ward's method [Fakes 92], [Manning 99].

Single-link is the simplest way with the smallest computational complexity among all, but the calculated value may not reflect the substantial relation between two clusters because it employ the smallest distance of the closest documents among two clusters. In case of the complete-link, the largest distance pair is used and computational complexity is largest among all. The rest two methods employ average distance of paired documents included in two clusters. Commonly used ways are the group-average link or Wards' method. Concerning with computational complexity, single-link is most easy.



$$S_C(C_a, C_b) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} S_D(d_{ai}, d_{bj})}{n \times m}$$

Figure 3-2: Illustration of Group-average link

3.3 Illustration of Hierarchical Clustering Results

The TRSM-based hierarchical clustering algorithm yields a list of generated clusters each of them consists of information about its father and sons, documents covered by the cluster, its dispersion (average of distances between all pairs of documents within the cluster), its representative. Below are some cluster representatives produced by the TRSM hierarchical clustering algorithm in form of a set of term-weight pairs.

Figure 3-3 shows an example cluster with parameters: θ = 6, distance calculation with upper approximations, and γ = 0.10. Here a cluster [CLS1035] consists of a cluster [CLS1033] and a cluster [MED_738] whose representatives are described by a set of hyphenated pairs of a term code and its weight. Figure 3-4 is the case of θ = 15. The cluster [CLS1035] consists of a cluster [MED_436] and a cluster [MED_424] with distance [0.44212]. These examples show that the difference of parameter θ makes the difference of not only cluster representatives but also the distance between clusters.

[0.64286] CLS1035=CLS1033+MED_738 kws:25+15-> 0;35;48 lvl 2

CLS1035: average distance = 0.187997 (3 docs)

------- clusters to be merged ----------

MED_738 : 9-0.121288, 126-0.241415, 244-0.294350, 260-0.300098, 458-0.214096, 607-0.322072, 629-0.228520, 909-0.335554, 1111-0.250418, 1112-0.147901, 1697-0.267799, 1922-0.267799, 3369-0.175716, 3613-0.297513, 4440-0.297513,

CLS1033: 109-0.124013, 123-0.125626, 240-0.140074, 244-0.237165, 250-0.237165, 291-0.145806, 303-0.145806, 402-0.152821, 431-0.157019, 458-0.157019, 547-0.161866, 607-0.283768, 629-0.283768, 785-0.174614, 909-0.295647, 957-0.183658, 1020-0.183658, 1185-0.183658, 1374-0.196406, 1945-0.196406, 2691-0.218198, 3615-0.218198, 3794-0.218198, 3875-0.218198, 4306-0.218198,

------- representatives of the new cluster ----------

REP CLS 1035: 9-0.071658, 109-0.099901, 123-0.101201, 126-0.101201, 240-0.112839, 244-0.236805, 250-0.191053, 260-0.115043, 291-0.117457, 303-0.117457, 402-0.123108, 431-0.126490, 458-0.214166, 547-0.130394, 607-0.283338, 629-0.283338, 785-0.140663, 909-0.295198, 957-0.147950, 1020-0.147950, 1111-0.147950, 1112-0.147950, 1185-0.147950, 1374-0.158219, 1697-0.158219, 1922-0.158219, 1945-0.158219, 2691-0.175774, 3369-0.175774, 3613-0.175774, 3615-0.175774, 3794-0.175774, 3875-0.175774, 4306-0.175774, 4440-0.175774,

------- upper approximation of the new cluster ----------

UPPER REP CLS 1035: 1-0.014457, 4-0.026338, 9-0.067372, 18-0.055419, 21-0.056598, 32-0.066503, 41-0.076003, 51-0.085810, 61-0.095004, 73-0.102312, 82-0.110838, 86-0.110838, 109-0.093926, 123-0.095148, 126-0.095148, 134-0.140006, 182-0.166257, 240-0.106090, 244-0.222642, 250-0.179626, 260-0.108162, 291-0.110431, 303-0.110431, 402-0.115745, 431-0.118924, 458-0.201357, 547-0.122595, 607-0.266391, 629-0.266391, 785-0.132250, 909-0.277542, 957-0.139100, 1020-0.139100, 1111-0.139100, 1112-0.139100, 1185-0.139100, 1374-0.148755, 1697-0.148755, 1922-0.148755, 1945-0.148755, 2691-0.165260, 3369-0.165260, 3613-0.165260, 3615-0.165260, 3794-0.165260, 3875-0.165260, 4306-0.165260, 4440-0.165260,

Figure 3-3: Cluster representatives with parameter $\theta = 6$

[0.44212] CLS1035=MED_436+MED_424 kws:15+15-> 0;22;23 lvl 1

CLS1035: average distance = 0.123319 (2 docs)

-------- clusters to be merged ----------

MED_436: 4-0.211185, 9-0.167962, 21-0.088018, 29-0.091972, 51-0.210091, 61-0.269007, 182-0.312394, 395-0.383863, 537-0.448479, 640-0.255316, 752-0.157106, 1170-0.279782, 1297-0.176713, 2947-0.332400, 3719-0.196321,

MED_424: 4-0.192811, 9-0.151590, 51-0.189612, 61-0.195257, 126-0.214084, 182-0.281050, 240-0.238705, 257-0.301647, 266-0.243367, 310-0.254118, 395-0.260429, 537-0.341899, 1170-0.312979, 3074-0.219615, 4021-0.371840,

------- representatives of the new cluster ----------

REP CLS 1035: 4-0.138383, 9-0.167676, 21-0.108911, 29-0.113803, 51-0.209734, 61-0.215978, 126-0.139860, 182-0.250812, 240-0.155944, 257-0.158990, 266-0.158990, 310-0.166014, 395-0.288066, 537-0.305115, 640-0.186587, 752-0.194398, 1170-0.346193, 1297-0.218659, 2947-0.242921, 3074-0.242921, 3719-0.242921, 4021-0.242921,

------- upper approximation of the new cluster ----------

UPPER REP CLS 1035: 4-0.138093, 9-0.167325, 20-0.064689, 21-0.108683, 29-0.113565, 51-0.209295, 61-0.215526, 126-0.139567, 182-0.250287, 240-0.155618, 257-0.158657, 266-0.158657, 310-0.165666, 395-0.287462, 537-0.304476, 640-0.186196, 752-0.193990,1170-0.345468, 1297-0.218201, 2947-0.242412, 3074-0.242412, 3719-0.242412, 4021-0.242412,

Figure 3-4: Cluster representatives with parameter $\theta = 15$

Chapter 4

Implementation

The algorithm described in chapter 3 is implemented and tested with six databases (see Section 5.1). The implementation of this clustering algorithm consists of two phases: one is the input preparation phase for the clustering with extracting the key terms of documents, generating tolerance classes of terms and approximations of documents, and the other is the clustering phase with preprocessed documents and related information.

4.1 Determination of the Rough Tolerance Space

As defined in Chapter 2, rough tolerance classes of terms are generated from the set of terms T using the term co-occurrence threshold θ. If the collection is made of the documents with full test, If the collection D consists of documents in their full text form, the process of keyword extraction is necessary before generating tolerance classes of terms and approximations of documents. Then, if the number of terms included in a document is somehow large, reducing them, as in most other methods, is significant for making rough tolerance spaces.

Thus preparation phase of generating rough tolerance spaces consists of two steps. The first step extracts keywords from the original text and calculates the importance of each term in a document. The second step provides inputs for the clustering procedure by

generating tolerance classes of terms and approximations of documents.

Figure 4-1 outlines the process flow of the preparation phase. It consists of the tasks:

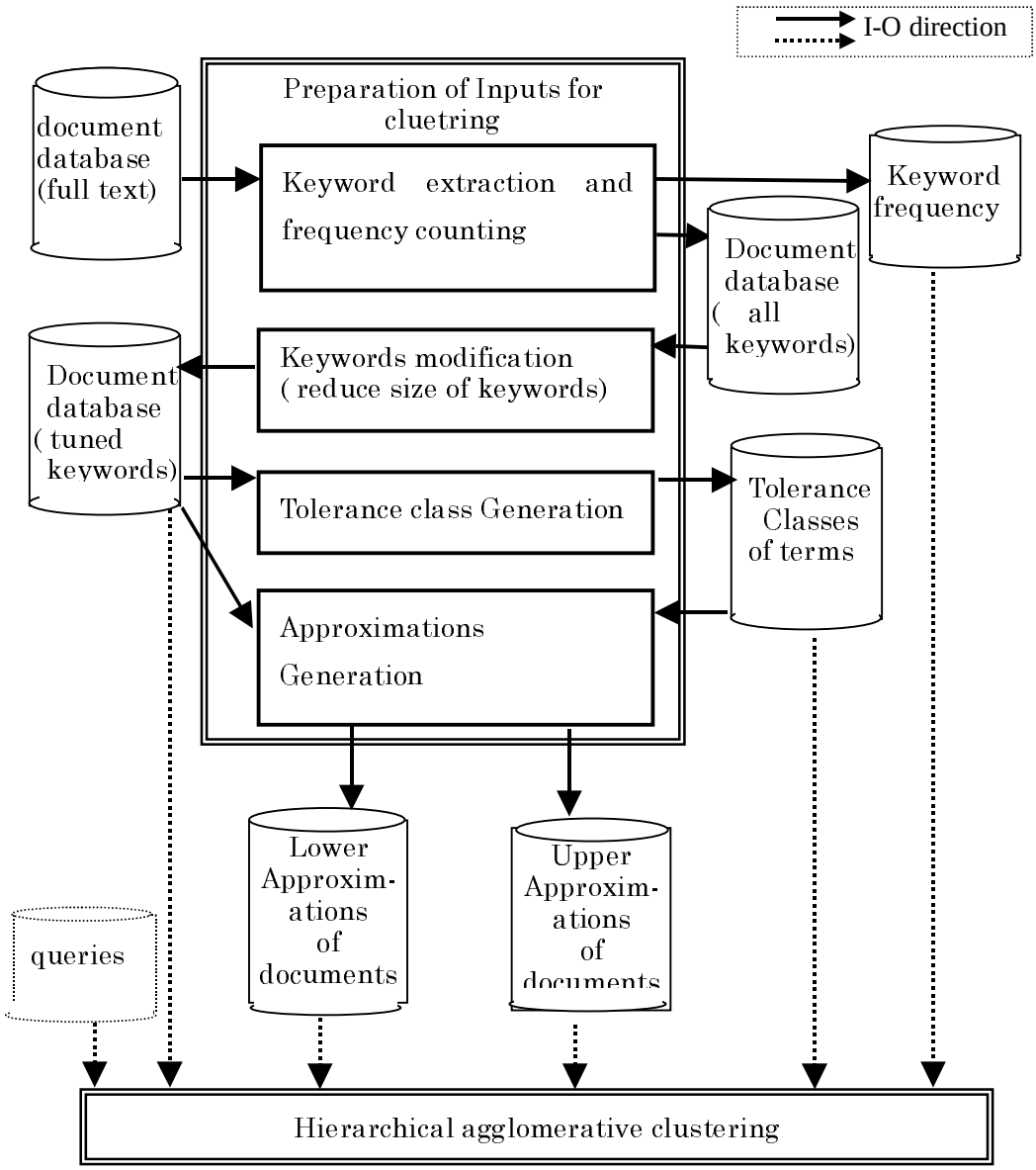


Figure 4-1: I-O flow of the preparation phase

1. **Keyword extraction** and **frequency counting**: inputs all documents of the specified database, counts frequencies of appeared terms and makes a new database of documents with sets of keywords and their frequencies.
2. **Keywords modification**: makes a database of documents with the specified number of keywords and their frequencies according to their $f_D(t_i)$, $f_{dj(ti)}$ values.
3. **Tolerance class generation**: generates a database of tolerance classes for terms according to the specified threshold of co-occurrence.
4. **Approximations generation**: generates an upper approximation database and a lower approximation database of documents according to the specified threshold of co-occurrence.

Since most of the processes concern with characters or strings, the programs are coded in Perl.

4.2 Clustering

Inputs for the clustering phase are a document database with cut-up keywords and their frequencies, a database of tolerance classes of terms, a database of keyword frequency and approximations of documents. According to the HACM algorithm described in Chapter 3 each document is regarded as a cluster on a leaf node, then distances between each paired in all combination of active clusters (an active cluster means the cluster which has not been merged into the more higher level cluster) are calculated using their keywords or approximations to find the most similar pair to merge next. There are three choices to calculate the distance between clusters: to compare document/cluster representatives, lower approximations and upper approximations of clusters. The program provides those three. In case of using original sets of keywords of documents, it is not different from the clustering by the basic vector space model.

After finding the most similar pair of clusters, they are merged into a new level of cluster. When creating a new node of the cluster, its representatives are calculated from representatives of merged clusters. The term weights of the new representatives are re-normalized. And weights of upper approximation are calculated as described chapter 3. Then distances between the new cluster and others are calculated. The cycle of this distance calculation, cluster merge and representatives calculation is repeated until all clusters are merged into one cluster. This part is implemented in language C. Data structure for the algorithm is shown in Figure 4-2. The fields for keywords and upper/lower approximations are allocated for pointers *kw, up, low* dynamically when the program runs because their occurrences depend on calculations. It is same as fields for term frequencies (*frq*) and term weightings of keywords and upper approximation (*weight*, *weight_up*). Each cluster points the addresses of its children nodes with pointers named *left* and *right*. The search actions for queries pass along with those pointers from the root node to the target nodes.

```
typedef struct db_list {     /* Database */
    char name[12];           /* Name of documents like <a10_10_10> */
                             /* List of Keywords */
    int *kw;                 /*[NORMAL_CLASS_SIZE];*/
    int size_kw;             /* size of keywords */
                             /* list of lower approximation */
    int *low;                /*[NORMAL_CLASS_SIZE];*/
    int size_low;            /* size of lower approximation */
                             /* list of upper approximation */
    int *up;                 /*[NORMAL_CLASS_SIZE];*/
    int size_up;             /* size of upper approximation */
    short *frq;              /* Frequency of terms in the doc */
    float *weight;           /* weight of terms in the doc */
    float *weight_up;        /* weight of upper-terms in the doc */
    int       level;
    double dis;              /* averaged distance representatives & documents*/
    struct db_list *next;
    struct db_list *left, *right;
} DB_LIST;
```

Figure 4-2: Data structure for documents, cluster and query

Figure 4-3 illustrates the outline of procedures for constructing the hierarchy of documents and a series of procedures for having precision and recall for queries.

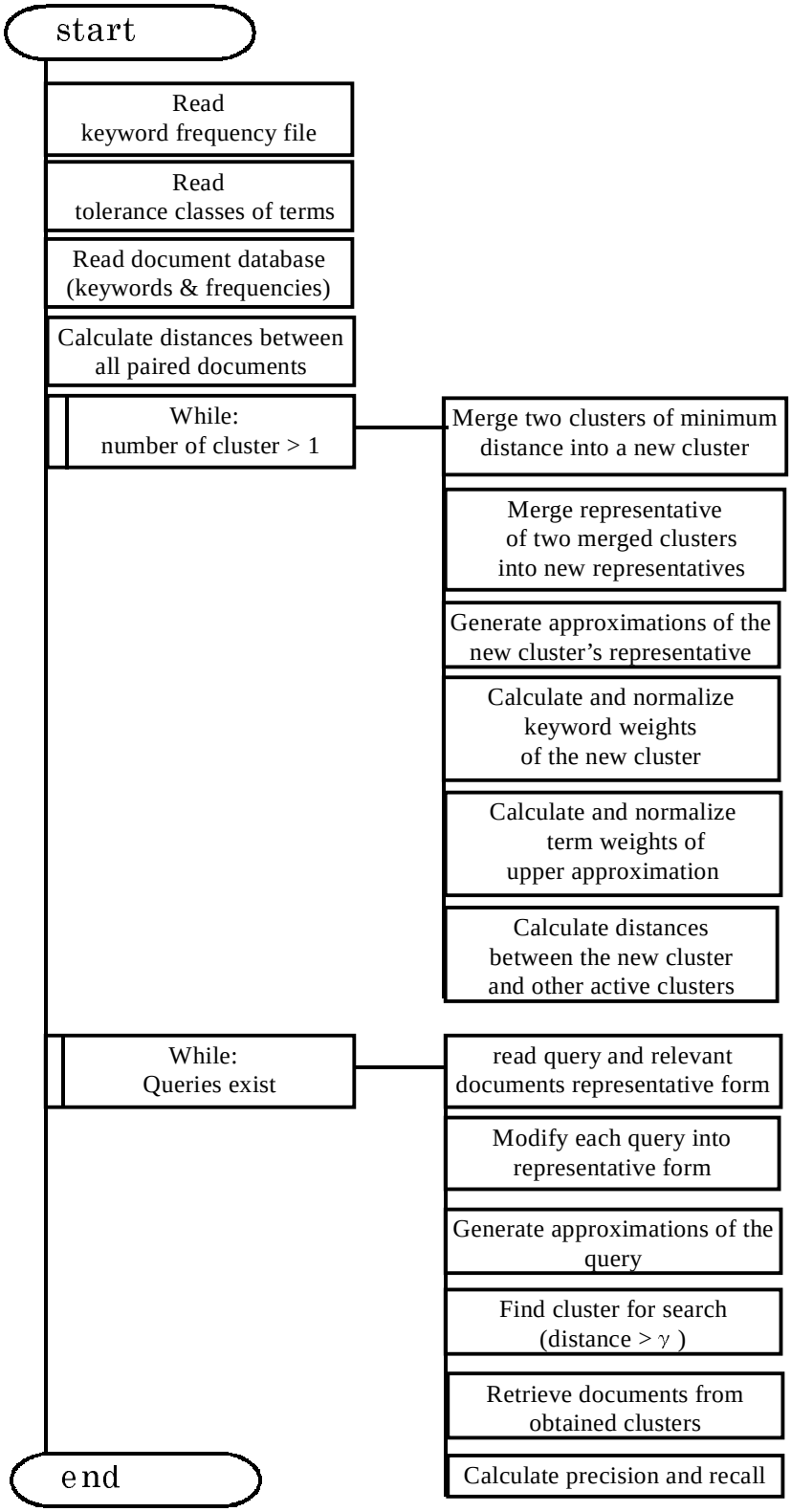


Figure 4-3: Procedure outline of TRSM-based hierarchical agglomerative clustering

4.3 Document Retrieval

In order to evaluate effectiveness of this algorithm, the queries processing for documents retrieval and calculation of precision and recall are also implemented in the clustering program. In this part, each given query with a set of keywords is transformed into the form of weighted terms as in representation of documents. After deciding the appropriate clusters using the distance between the query and cluster representatives, documents are retrieved and compared with relevant documents predefined for the query. The retrieving strategy through the hierarchy consists of the following steps.

1. From the root nodes of the hierarchy, the distances between the query and cluster representatives are calculated by top-down direction to find the appropriate clusters as target clusters to search. The condition of an appropriate cluster is that the distance between the query and representatives of the clusters is less than the threshold γ. Once to have a cluster, lower levels of it are not necessary to be checked any more.
2. After having the target clusters, all documents included in the clusters are checked their distances from the query and similar documents are retrieved.
3. Compare the retrieved documents and relevant documents, then we have the values of precision and recall.

The evaluation result will be discussed in next chapter.

Chapter 5

Evaluation and Validation

Evaluation and validation are important issues in clustering [Gordon 96]. A clustering method is often evaluated in terms of its performance in doing specified tasks. Cluster validation procedures aim to verify whether the data structure produced by the clustering method can be used to provide statistical evidence of the phenomenon under study [Willet 88], [Fakes 92], [Baeza-Yates 99].

In this chapter, the test collections used in experiments are presented first, then the results of typical kinds of evaluation and validation of the method follows. The evaluation focuses on the efficiency and effectiveness in information retrieval with and without TRSM-based clustering. The validation focuses on the tendency and stability of the TRSM-based clustering results.

5.1 Test Collections

The TRSM-based hierarchical clustering method and its implemented programs have been tested with 6 document collections, including JSAI [Ho 98], CACM, CISI, MED, CRAN [Fox 90], and Reuters [Yang 99]. All of collections used here are English. Table 5.1 shows information of these test collections.

The JSAI collection contains papers and articles of the Journal of Japanese Society for Artificial Intelligence during 1986-1995. Each document in this collection consists of not

only author name, paper title, published year, volume, number and pages, but also a set of keywords provided by the author. No need of pre-processing (first phase) for JSAI, but also there are no weights for keywords. It is a special kind of text databases in information retrieval.

Table 5-1: Test collections ("*" stands for the number of categories)

Name of collection	Subject of the collection	Number of documents	Number of terms	Number of queries	Number of relevant doc.
JSAI	Artificial Intelligence	802	1813	20	32
CACM	Computer Science	3200	6520	64	15
CISI	Library	1460	4414	76	40
MED	Medicine	1033	4841	30	23
CRAN	Aeronautics	1398	3182	225	8
Reuters v3	Newswire stories	9610	15550	93 (*)	
Reuters v4	Newswire stories	7789	24933	93 (*)	

The CACM, CISI, MED and CRAN are well-known test collections in the field of information retrieval. Each of them is associated with a set of queries and for each query a set of relevant documents from the collection [Fox 90].

The Reuters is a well-known test collection used in text categorization. It consists of training and testing documents that are labeled documents (supervised data). There are different versions of the Reuters corpus. It is employed here the Reuters version 3 (provided by the research group at Carnegie Melon University) and version 4 (provided by the research group at Xerox PARC) [Yang 99]. However, as clustering techniques aim to find "natural" clusters from unsupervised data, we will mainly focus on test collection for information retrieval.

All of these test collections require the preprocessing. As mentioned previously, in each collection and each document, there are many terms that have same meaning, or the same stem. They are modified to merge together as the same terms during the preprocessing phase.

The clustering quality for each test collection depends on parameter θ in TRSM and on σ in the clustering algorithm. It is important to 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 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 5-3 of section 5-2 we can see how retrieval effectiveness relates to different values of θ. To avoid biased experiments when comparing algorithms, we take the common default values $\theta = 15$ and $\sigma = 0.1$ for all test collections.

5.2 Evaluation

5.2.1 Evaluation of Cluster-based Retrieval Effectiveness

In case of information retrieval and in text processing in general, the performance of a system is often evaluated in terms of *efficiency* and *effectiveness*, the two measures how well the system satisfies its user.

Efficiency indicates time or cost until having results for the request. *Effectiveness* is an indication of preciseness and completeness of the answer for the user. The preciseness and completeness are evaluated by using measures of recall and precision.

Recall is the fraction of the relevant documents that has been retrieved. *Precision* is the fraction of the retrieved documents that is relevant. Recall and precision are the most basic measures for evaluation in information retrieval and text processing [Baeza-Yates, 99]. Assume that a set of queries got documents from a collection. The set of documents retrieved for the queries contains both of relevant to the queries and non-relevant to the queries. Also the rest set of documents not retrieved contains both. Thus the whole set of documents in a collection can be divided into four classes with each numbers as *w, x, y* and *z* in Figure 5-1.

	Number of retrieved	Number of not-retrieved
Number of relevant	w	x
Number of non-relevant	y	z

Figure 5-1: Class matrix of documents

According to this notion in [徳永, 99], precision and recall can be expressed as follows:

$$\text{Recall} \qquad R = \frac{w}{w+x}$$

$$\text{Precision} \qquad P = \frac{w}{w+y}$$

Both precision and recall have the value range of [0,1]. It is a usual trend that when precision is high, recall may be low and vice-versa. For example, if all documents in a set are retrieved, because it means all documents relevant to the query are included, the recall becomes maximum value 1.

When using a clustered database, the efficiency should be better than when using the whole database because the target space for a certain aim is able to be small, while the preparation for clustered database requires time. It is also predicted that even if efficiency is well, effectiveness is not so bad or increase when the clustering is well done [Iwayama 95]. Because, by nature, clustering is the kind of method which groups objects with similar characteristics and in case of finding appropriate clusters for the query.

Table 5-2 shows a comparison on precision and recall of the TRSM-based full retrieval and the VSM-based full retrieval (Vector Space Model) 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.

Table 5-2: Precision and recall of full retrieval based on TRSM and VSM

	JSAI		CACM		CISI		CRAN		MED	
θ	*P*	*R*	*P*	*R*	*P*	*R*	*R*	*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	.0141	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**

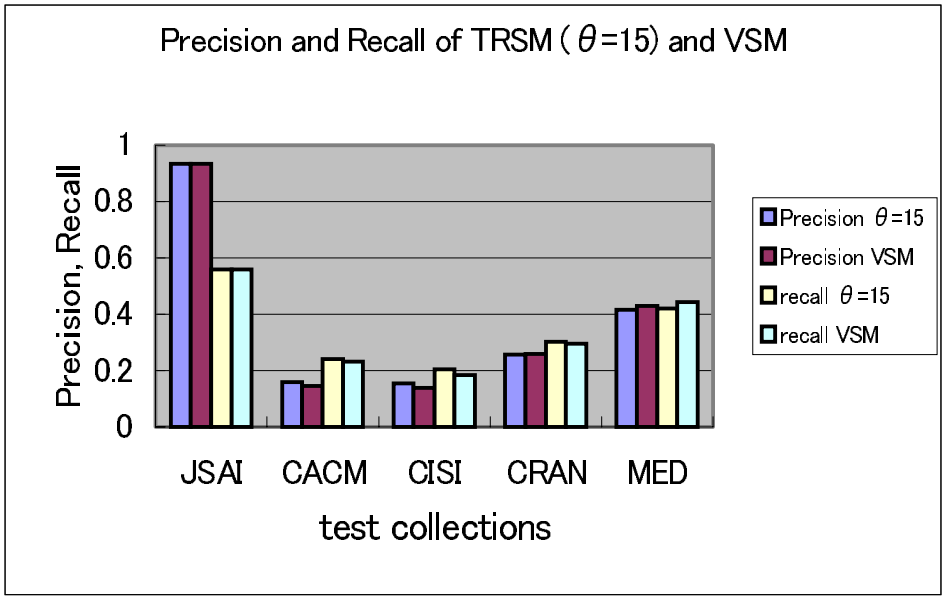


Figure 5-2: Precision and recall of TRSM (θ =15) and VSM

From Table 5-2 we see that precision and recall for JSAI are high, and they are higher and almost 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 original VSM. Figure 5-2 gives a graphical view between these two methods with θ = 15 for TRSM-based method.

It was 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 $D' \subseteq D$. The subset D' is the 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 D'.

Table 5-3 reports the average of precision and recall for all queries in test collections using the TRSM cluster-based retrieval with various proportion (%) of D' to D, and full retrieval in the whole D (accordingly, values of γ. The results show that in several cases (JSAI, CISI, and MED) just searching a small part of 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 $|D'|$ is about 17% of $|D|$. More importantly, the TRSM cluster-based search offers precision higher than that of full retrieval in many collections. Also, the TRSM cluster-based retrieval achieved recall and precision nearly as that of full search just after searching a small portion of D.

Table 5-4 reports the effectiveness of TRSM cluster-based retrieval (TRSM) versus VSM cluster-based retrieval (VSM) when $|D'|$ is 2.9%, 8.0%, and 16.9% of $|D|$. It shows that TRSM cluster-based retrieval often achieves precision higher than that of VSM cluster-based retrieval thought its recall is somehow lower.

These results suggest that TRSM can be used to improve precision of information retrieval, and so in a certain tasks of text mining.

Table 5-3: Precision and recall of the TRSM cluster-based and full search

Collection	1.2% (0.18)		1.8% (0.16)		2.9% (0.14)		8.0% (0.11)		16.9% (0.09)		Full search	
	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

Table 5-4: Precision and recall of the TRSM and VSM cluster-based retrieval

	2.9% of D (γ = 0.14)				8.0% of D (γ = 0.11)				16.9% of D (γ = 0.09)			
	TRSM		VSM		TRSM		VSM		TRSM		VSM	
	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	0.1994	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

5.2.2 Evaluation of TRSM Hierarchical Clustering Efficiency

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, 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 $O(M+N)$.

All the experiments reported in this paper were performed on a conventional workstation

GP7000S Model 45 (Fujitsu, 250 MHz Ultra SPARC-II, 512 MB). Note that the search for clusters requires in average log M, and 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 5-5 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 proposed TRSM-based method is applicable to large document collections.

Table 5-5: Performance Measurements of the TRSM Cluster-based Retrieval

Collection	Size (MB)	Number of queries	TRSM Time	Clustering Time	Full Search (sec)	Cluster Search (sec)	Memory (MB)
JSAI	0.1	20	14.9s	14.9s	0.8	0.1	8
CACM	2.2	64	22m2.8s	26m46.8s	13.3	1.2	201
CISI	2.2	76	13m16.8s	4m49.8s	40.1	3.4	84
CRAN	2.6	225	23m9.9s	3m6.9s	20.5	1.8	71
MED	1.1	30	40.1s	1m30.8s	2.5	0.3	25
Reuters	5.0	n/a	16m42.3s	173m	n/a	n/a	820

5.2.3 Text Categorization

For evaluating how well the TRSM clustering work for text categorization, the Reuters collection is used, because it is the most commonly used collection for text categorization evaluation in the literature [Yang 99]. In case of category ranking with machine learning algorithms, the measures for evaluation are precision and recall like in document retrieval. The definitions of those measures here are:

recall = categories found and correct / total categories correct
precision = categories found and correct / total categories correct

where “categories found” means categories above the decision threshold.

However, in case of the proposed TRSM without learning algorithm, as a somehow deterministic approach for the dataset, it is not able to say whether clusters located at certain levels reflect their corresponding categories. Therefore, firstly, the relationship between clusters and categories are checked. An assumption is applied that, in the hierarchy of documents, the nodes are regarded as homogeneous when there is a large proportion (regarding some threshold δ, for example 0.5) of documents belonging to one category and the upper nodes do not satisfy the condition. An additional condition is the averaged distance between the representatives of each cluster and included documents should be has less than some other threshold. The clustering results reflecting the assumption made clusters labeled by categories as shown in Table 5-6.
Figure 5-3 illustrates the numbers of clusters which includes 1, 2-5, 6-10, …, and more than 1000 documents for each condition. According to the given condition, more than half numbers of clusters consist of only one document, and other clusters include less than 21 documents close to the averages, but categories representing them are sometimes same so that documents belong to other categories are dispersed into different clusters.

Table 5-6: The homogeneity of generated clusters

	Number of cluster (averaged distance)					
	$\delta = 0.5$		$\delta = 0.4$		$\delta = 0.3$	
	cluster	category	cluster	category	cluster	category
Reuters version 3	**1273**	**62(17)**	**493**	**46(26)**	**294**	**37(30)**
Reuters version 4	**478**	**44(15)**	**936**	**57(18)**	**801**	**51(38)**

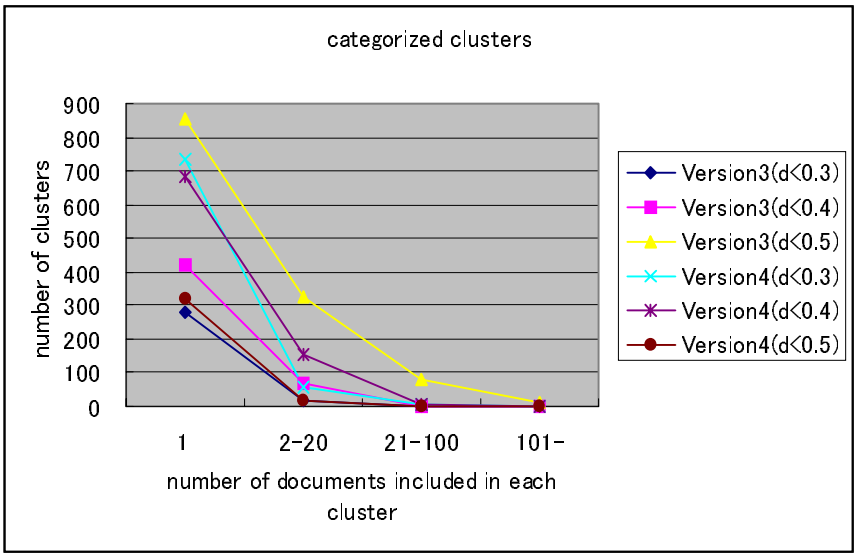


Figure 5-3: Documents distribution of categorized clusters

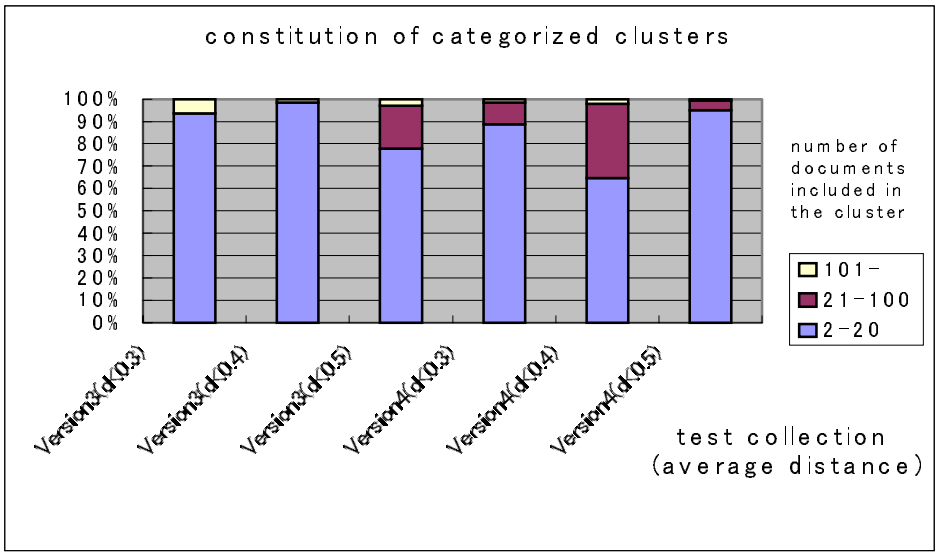


Figure 5-4: Number of documents included in categorized clusters

As a result, only a few clusters can represent corresponding categories. It depends on which dataset to use because this algorithm groups related documents according to their keywords and co-occurrences of keywords.

5.3 Validation

The result of clustering is validated whether it is a valid summary of the documents or whether unwarranted or inappropriate structure is being imposed on the collection.

The extent of support for hierarchical structure in a data set has also been assessed by stability. For the validation, five collections except Reuters are used because these validation measures need to use the relevant documents.

5.3.1 Clustering Tendency

The experiments attempt to determine whether worthwhile retrieval performance would be achieved by clustering a database, before investing the computational resources which clustering the database would entail [Fakes 92], [Willet 88].

The *nearest neighbor test* [Willet 88] is employed here 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.

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. Data collections used for validation of clustering tendency are described in Table 5-1. Columns 4 and 5 show the number of queries and total number of relevant documents for all queries in each test collection. Table 5-6 reports the experimental results synthesized from those done on five test collections. The each rows with numbered as 0, 1, 2, 3, 4 and 5 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 6 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 is appropriate for the retrieval purpose. This tendency can be clearly observed in concordance with the high retrieval effectiveness for JSAI and MED shown in Table 5-7 and Figure 5-5.

Table 5-7: Results of clustering tendency

Test collection	Average percentage of relevant documents (%)						Ave.
	0	1	2	3	4	5	
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

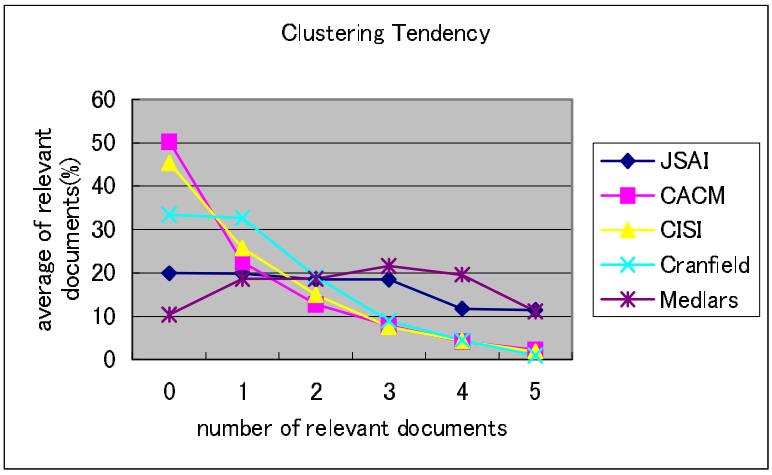


Figure 5-5: Clustering tendency

5.3.2 Clustering stability

Clustering stability is an assessment that the original classifications are compared with classifications of modified versions of the data. If there is a large difference between those classifications, the clustering is considered unstable.

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.

Table 5-8 and Figure 5-6 show the experimental results of clustering stability for JSAI test collection with different values of *s* from 210 experiments with *s*% = 1%, 2%, 3%, 4%, 5%, 10% and 15%. 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 *D* [Willet 88]. The way is, first, to remove randomly a specified amount of *s*% documents from the collection, then re-determine the new tolerance space for the reduced database to perform the TRSM clustering algorithm and evaluate the change of clusters due to the change of the database. 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 for other test collections have nearly the same results. It suggests that the TRSM hierarchical clustering is stable.

Table 5-8: Synthesized results about the stability

θ	Percentage of cluster stability (%)						
	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

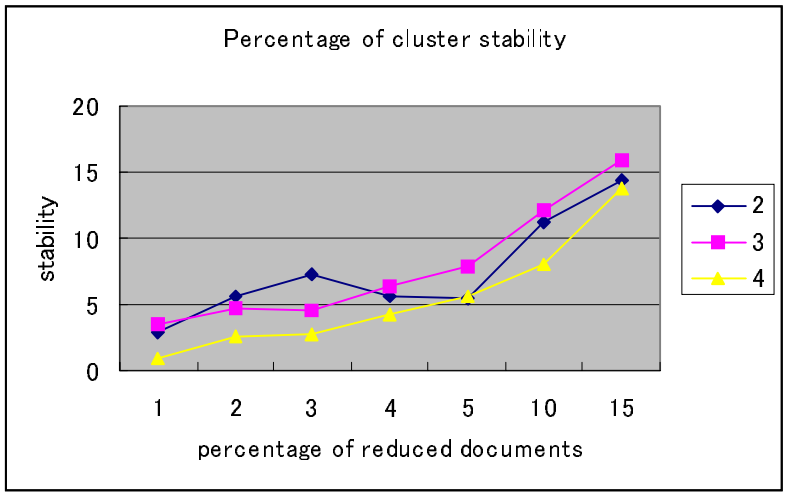


Figure 5-6: Synthesized results about the stability

Chapter 6
Conclusion

This paper has presented a TRSM-based clustering algorithm within the HACM framework. TRSM is a tool that offers a new method of semantics calculations in text processing. In the framework of TRSM, this algorithm has achieved relatively remarkable results on its effectiveness, efficiency and validity thank to finding some kinds of semantic relations that do not directly appear in the textual database. The representation of documents and clusters enriched by not only normalized term weighting but also by approximations allows us to calculate exquisite distances. The experimental results suggest that this algorithm can be applied to the field of text processing. However, in order to apply this algorithm to text categorization, as it cannot show any appealing result, there is a room for more consideration of improvement on the method. About efficiency, the implementation has achieved an efficient improvement on the processing time, while this method has the order of N square computational complexity depending on the number of terms and document and the trial implementation shows the memory consuming of nearly 1giga bytes for 15550 terms in about 10000 documents. It means that it is difficult to apply this hierarchical clustering algorithm to much larger databases.

According to such kinds of limitations, further investigation of this method can be able to be considered on the following directions:

- Introducing similarity relations instead of tolerance relations [Skowron 97]. Similarity relation does not require the reflexive property for the classes in a set so that it seems to introduce more flexible semantic relations into the model.

- Using lower approximations in the part acquired by upper approximations may reflect the strength of term relationship more precisely.

- Development of a hybrid method that combines hierarchical clustering and non-hierarchical clustering for databases of large size. The merit of non-hierarchical clustering method is that it requires much less computational time and space than any hierarchical clustering method. A considerable way for dealing with very large text collection consists of two parts, i.e., at first the collection is divided into a certain numbers of clusters by the non-hierarchical clustering based on TRSM, then TRSM based hierarchical clustering constructs the hierarchical structure for of each obtained cluster.

For practical use, this method should be link to other methods appropriate for user purposes, especially concerning with user interfaces to visualize of results and procedures of a series of interactive procedures between users and machines.

Acknowledgements

First of all, I would like to express profoundly my appreciation to Professor Tu Bao Ho for his a lot of kindness, patience and effectual support during the work. I would not be able to complete this thesis without his advice and encouragement. Then I appreciate a lot for Dr. Ngoc Binh Nguyen, our previous Associate, for helping me a lot on programming. My program would not run without him. Mr. Kaname Funakoshi's work has suggested so many things that I am obliged him even if I have never seen him.

And, I profoundly thank Associate Professor Masato Ishizaki in Knowledge Creating Laboratory for his pertinent and sound comments and advice about natural language processing, and Professor Yoshiteru Nakamori for his warm help. Then thanks a lot for Saitou-san and Fujikawa-san, member of the same laboratory, for sharing daily activities with friendship.

Since I like lectures here which I took as my required course works, I would like to appreciate all of them and people who provided them for introducing me basics and ideas which were new and fresh for an ex-working women.

Next thank is for people who were assigned temporary on 7th floor and for other friends in JAIST in sharing with me a jolly college life with good dishes.

Finally, I would like to thank sincerely my father, mother, grandmother and sisters not only to allow me to stay and study far from them after a long absence of my previous student life and jobs, but also to have given me a lot of supports in many aspects.

August, 2000

Bibliography

[Baeza-Yates 99] Baeza-Yates, R. and Ribeiro-Neto, B., *Modern Information Retrieva*l, Addison Wesley, 1999.

[Boyce 94] Boyce, B. R., Meadow, C. T., and Donald, H. K., *Measurement in Information Science*, Academic Press, 1994.

[Fakes 92] Fakes, W. B. and Baeza-Yates, *Information Retrieval. Data Structures and Algorithms* (Eds.), Prentice Hall, 1992.

[Fox 90] Fox, E., *Virginia Disk One* (Ed.), Blacksburg: Virginia Polytechnic Institute and State University, 1990

[Gordon, 96] Gordon, A. D., "Hierarchical Classification", *Clustering and Classification*, World Scientific, 1996, 65-121.

[Ho 98] Ho, T. B. and Funakoshi K., "Information retrieval using rough sets", *Journal of Japanese Society for Artificial Intelligence*, Vol. 13, N. 3, 1998, 424-433.

[Iwayama 95] Iwayama and M, Tokunaga, T., "Hierarchical Bayesian Clustering for Automatic Text Classification", *International Joint Conference on Artificial Intelligence IJCAI'95*, Morgan Kaufmann, 1995, 1322-1327.

[Kawasaki 00] Kawasaki, S, Nguyen, N.B. and Ho, T.B., "Hierarchical Document Clustering Based on Tolerance Rough Set Model", *4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Lyon, France, September 13-16, 2000. Lecture Notes in Artificial Intelligence, Springer, 2000, 6 pages.

[Larsen 99] Larsen, B. and Aone, C., “Fast and effective text mining using linear-time document clustering”, *International Conference on Knowledge Discovery and Data Mining KDD'99*, Morgan Kaufmann, 1999, 16-22.

[Lebart 98] Lebart, L., Salem, A., and Berry, L., *Exploring Textual Data*, Kluwer Academic Publishers, 1998.

[Lin 97] Lin, T. Y. and Cercone, N., *Rough Sets and Data Mining. Analysis of Imprecise Data* (Eds.), Kluwer Academic Publishers, 1997.

[Manning 99] Manning, C. D. and Schutze, H., *Foundations of Statistical Natural Language Processing*, The MIT Press, 1999.

[Miyamoto 90] Miyamoto, S., *Fuzzy Sets in Information Retrieval and Cluster Analysis*, Kluwer Academic Publishers, 1990.

[Pawlak 91] Pawlak, Z., *Rough sets: Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, 1991.

[Polkowski 98] Polkowski, L. and Skowron, A., *Rough Sets in Knowledge Discovery. Applications, Case Studies and Software Systems* (Eds.), Physica-Verlag, 1998.

[Raghavan 86] 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, 1986, 88-101.

[Salton 88] Salton, G. and Buckley, C., “Term-Weighting Approaches in Automatic Text Retrieval”, *Information Processing & Management*, Vol. 24, N. 5, 1988, 513-523.

[Sharma 96] Sharma, S., *Applied Multivariate Techniques*, John Wiley & Sons, Inc., 1996.

[Skowron 94] Skowron, A. and Stepaniuk, J., “Generalized approximation spaces”, *The 3rd International Workshop on Rough Sets and Soft Computing RSKD’93*, Springer-Verlag, 1994, 156-163.

[Skowron 97] Skowron, A. and Vanderpooten, D., “Similarity Relation as a Basis for Rough Approximations”, *Advances in Machine Intelligence and Soft Computing*, P. Wang (Ed.), Vol. 4, 1997, 17-33

[Srinivasan 89] Srinivasan, P., “Intelligent Information Retrieval Using Rough Set Approximations”, *Information Processing and Management*, Vol. 25, No.4, 1989, 347-361.

[Srinivasan 91] Srinivasan, P., “The Importance of Rough Approximations for Information Retrieval, *International Journal of Man-Machine Studies*, Vol. 34, No. 5, 1991, 657-671.

[Van Rijsbergen 77] Van Rijsbergen, C.J.: “A Theoretical Basis for the Use of Co-occurrence Data in Information Retrieval”, *Journal of Documentation*, Vol. 33, No. 2, 1977, 106-119.

[Willet 88] Willet, P., “Recent trends in hierarchical document clustering: A critical review”, *Information Processing and Management*, 1988, 577-597.

[Yang 99] Yang, Y., “An evaluation of Statistical Approaches to Text Categorization”, *Information Retrieval Journal*, Kluwer Academic Publishers, 1999, 1-19.

[石岡 98] 石岡恒憲・亀田雅之, 単語の共起に基く関連文書検索、算法と検索事例, 応用統計学 Vol. 28, No. 2, 1998, 107-121.

[笠井 99] 笠井透・板井力・有村博紀・有川節夫, テキストデータマイニングに基く探索的文書ブラウジング, 日本ソフトウェア科学会データマイニングワークショップ, 1999.

[関根聡 99] 関根聡, テキストからの情報抽出－文書から特定の情報を抜き出す－, 情報処理, 40-4,1999, 370-373.

[中村 96] 中村昭・津本周作・田中博・小林聡, ラフ集合とその応用, 人工知能学会誌 Vol. 11, No. 2, 1996, 209-215.

[那須川 99] 那須川哲哉・諸橋正幸・長野徹, テキストマイニング－膨大な文書 データの自動分析による知識発見－, 情報処理, 40-4,1999, 358-364.

[藤田 99] 藤田澄男, 自然言語処理を利用した情報の検索・分類へのアプローチ, 情報処理, 40-4,1999, 352-357.

Contributions

[1] Kawasaki, S., Nguyen, N. B. and Ho, T. B., "Hierarchical Document Clustering Based on Tolerance Rough Set Model", *The Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases* (PKDD2000), Lyon, France, September 13-16, 2000.

Appendix.

Programs for TRSM-based hierarchical clustering (basic version)