Rough Document Clustering and The Internet

Hung Son Nguyen
Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
Tu Bao Ho
Japan Advanced Institute of Science and Technology,
Tatsunokuchi, Ishikawa, Japan

No Institute Given

Abstract. Searching for information on the web has attracted many research communities. Due to the enormous size of the web and low precision of user queries, finding the right information from the web is the difficult or even impossible task. Clustering, one of the most the fundamental tools in Granular Computing (GrC), offers an interesting approach to this problem. By grouping of similar documents, clustering approach facilitates the presentation of search results in more compact form and enable thematic browsing of the results set. The main problem of existing web search result (snippet) clustering algorithms is based on the poor vector representation of snippets. In this paper, we present a method of snippet representation enrichment based on the rough set theory. We apply the Tolerance Rough Set (TRS) model for document collection to construct a rough set based search result clustering algorithm and compared it with other recent methods.

Keywords: rough sets, granular computing, snippet, clustering.

1 Introduction

Granular Computing (GrC) was characterized as a common name of theories, methodologies, techniques, and tools that make use of granules (groups, classes or clusters) in the process of problem solving [2]. From this point of view clustering can be treated as an information granulation process, which is also the main step of Computing with Words (CW) approach.

Rough set theory has been introduced by Pawlak [11] as a tool for concept approximation under uncertainty. The idea is to approximate the concept by two descriptive sets called *lower and upper approximations*. The main philosophy of

rough set approach to concept approximation problem is based on minimizing the difference between upper and lower approximations (also called the *boundary region*). This simple, but brilliant idea, leads to many efficient applications of rough sets in machine learning, data mining and also in granular computing.

The connection between rough set theory and granular computing was examined by many researchers. In [12] [13] [7] [29] some particular semantics and interpretations of information granules were defined, and some algorithms for constructing granules were given. Many clustering methods based on rough sets and other computational intelligence techniques were proposed including support vector machine (SVM) [24], genetic algorithm (GA) [26] [30], modified self-organizing map (SOM) [28]. The rough set based clustering methods were applied to many real life applications, e.g., medicine [25], web user clustering [27] [28] and marketing [30].

This paper presents the rough set approach to document clustering and its application in search engine technology, particularly, in the web search result clustering problem. Let us explain the problem more precisely and present its current state of the art.

Two most popular approaches to facilitate searching for information on the web are represented by web search engine and web directories. Web search engines [1] allow user to formulate a query, to which it responds using its index to return set of references to relevant web documents (web pages). Web directories [2] are human-made collection of references to web documents organized as hierarchical structure of categories.

Although the performance of search engines is improving every day, searching on the web can be a tedious and time-consuming task because (1) search engines can only index a part of the "indexable web", due to the huge size and highly dynamic nature of the web, and (2) the user's "intention behind the search" is not clearly

[1] e.g., Altavista (http://www.altavista.com), AllTheWeb (http://www.alltheweb.com), Google(http://www.google.com), HotBot (http://www.hotbot.com), Lycos (http://www.lycos.com),
[2] e.g., Yahoo (http://www.yahoo.com) or Open Directory Project (http://www.dmoz.org)

expressed by too general, short queries. As the effects of the two above, results returned by search engine can count from hundreds to hundreds of thousands of documents.

One approach to manage the large number of results is clustering. The concept arises from document clustering in Information Retrieval domain: *find a grouping for a set of documents so that documents belonging to the same cluster are similar and documents belonging to different cluster are dissimilar*. Search results clustering thus can be defined as *a process of automatical grouping search results into to thematic groups*. However, in contrast to traditional document clustering, clustering of search results are done on-the-fly and locally on a limited set of results return from the search engine. Clustering of search results can help user navigate through large set of documents more efficiently. By providing concise, accurate description of clusters, it lets user localizes interesting document faster.

Despite being derived from document clustering, methods for clustering web search results differ from its ancestors on numerous aspects. Most notably, document clustering algorithms are designed to works on relatively large collection of full-text document (or sometimes document abstract). In opposite, the algorithm for web search results clustering is supposed to works on moderate size (several hundreds elements) set of text snippets (with length of 10-20 words).

In document clustering, the main emphasis is put on the quality of clusters and the scalability to large number of documents, as it is usually used to process the whole document collection (e.g. for document retrieval on clustered collection). For web search results clustering, apart from delivering good quality clusters, it is also required to produce meaningful, concise description for cluster. Additionally, the algorithm must be extremely fast to process results on-line (as post-processing search results before delivered to the user) and scalability with the increase of user

requests. (e.g. measures as number of processed requests per specified amount of time).

	Document clustering	**Web Search result clustering**
Objects	full-text documents or document abstracts	short text snippets
Processing mode	off-line processing on a large collection	on-line processing of moderate size set of snippets
Quality measurement	cluster quality	cluster quality and cluster description meaningfulness
Computation requirement	scalability with number of documents	scalability with number of user requests

The earliest work on clustering results were done by Pedersen, Hearst et al. on Scather/Gather system [3], followed with application to web documents and search results by Zamir et al. [23] to create Grouper based on novel algorithm Suffix Tree Clustering. Inspired by their work, a Carrot framework was created by Weiss [20] to facilitate research on clustering search results. This has encouraged others to contribute new clustering algorithms under the Carrot framework like LINGO [8][10], AHC [22].

In this paper, we proposed an approach to search results clustering based on Tolerance Rough Set following the work on document clustering of Bao [16, 19]. The main problem that occurs in all mentioned works is based on the fact that many snippets remain unrelated because of their short representation (see Table 3). Tolerance classes are used to approximate concepts existed in documents and to enrich the vector representation of snippets. Set of documents sharing similar

concepts are grouped together to form clusters. Concise, intelligible cluster labels are next derived from tolerance classes using special heuristic.

2 Rough Sets and Tolerance Rough Set Model

Rough set theory was originally developed [11, 5] as a tool for data analysis and classification. It has been successfully applied in various tasks, such as feature selection/extraction, rule synthesis and classification [5]. In this chapter we will present fundamental concepts of rough sets with illustrative examples. Some extensions of rough set are described, concentrating on the use of rough set to synthesize approximations of concepts from data.

Consider a non-empty set of objects U called the universe. Suppose we want to define a concept over universe of objects U. Let us assume that our concept can be represented as subset X of U. The central point of rough set theory is the notion of set approximation: any set in U can be approximated by its *lower* and *upper approximation*.

2.1 Generalized approximation spaces

The classical rough set theory is based on equivalence relation that divides the universe of objects into disjoint classes. By definition, an equivalence relation $R \subseteq U \times U$ is required to be reflexive, symmetric, and transitive. Practically, for some applications, the requirement for equivalent relation has showed to be too strict. The nature of the concepts in many domains are imprecise and can be overlapped additionally. For example, let us consider a collection of scientific documents and keywords describing those documents. It is clear that each document can have several keywords and a keyword can be associated with many documents. Thus, in the universe of documents, keywords can form overlapping classes.

Skowron [17] has introduced a generalized tolerance space by relaxing the relation R to a tolerance relation, where transitivity property is not required. Formally, the generalized approximation space is defined as a quadruple $\mathcal{A} = (U, I, \nu, P)$, where

- U is a non-empty universe of objects,
- $I : U \rightarrow \mathcal{P}(U)$, where $\mathcal{P}(U)$ is a power set of U, is an *uncertainty function* satisfying conditions: (1) $x \in I(x)$ for $x \in U$, and (2) $y \in I(x) \iff x \in I(y)$ for any $x, y \in U$. Thus the relation $xRy \iff y \in I(x)$ is a tolerance relation and $I(x)$ is a tolerance class of x,
- $\nu : \mathcal{P}(U) \times \mathcal{P}(U) \rightarrow [0, 1]$ is a *vague inclusion function*. Vague inclusion ν measures the degree of inclusion between two sets. Vague inclusion must be *monotone* with respect to the second argument, i.e., if $Y \subseteq Z$ then $\nu(X, Y) \leq \nu(X, Z)$ for $X, Y, Z \subseteq U$.
- $P : I(U) \rightarrow \{0, 1\}$ is a *structurality function*.

Together with uncertainty function I, vague inclusion function ν defines the *rough membership function* for $x \in U, X \subseteq U$ by $\mu_{I,\nu}(x, X) = \nu(I(x), X)$. Lower and upper approximations of any $X \subseteq U$ in $\mathcal{A}$, denoted by $\mathbf{L}_{\mathcal{A}}(X)$ and $\mathbf{U}_{\mathcal{A}}(X)$, are respectively defined as

$$\mathbf{L}_{\mathcal{A}}(X) = \{x \in U : P(I(x)) = 1 \wedge \nu(I(x), X) = 1\}$$
$$\mathbf{U}_{\mathcal{A}}(X) = \{x \in U : P(I(x)) = 1 \wedge \nu(I(x), X) > 0\}$$

With given definition above, generalized approximation spaces can be used in any application where I, ν and P are appropriately determined.

2.2 Tolerance Rough Set Model

Tolerance Rough Set Model (TRSM) was developed [16, 19] as basis to model documents and terms in information retrieval, text mining, etc. With its ability to deal

with vagueness and fuzziness, tolerance rough set seems to be promising tool to model relations between terms and documents. In many information retrieval problems, especially in document clustering, defining the similarity relation between document-document, term-term or term-document is essential. In Vector Space Model, is has been noticed [19] that a single document is usually represented by relatively few terms. This results in zero-valued similarities which decreases quality of clustering. The application of TRSM in document clustering was proposed as a way to enrich document and cluster representation with the hope of increasing clustering performance.

The idea is to capture conceptually related index terms into classes. For this purpose, the tolerance relation R is determined as the co-occurrence of index terms in all documents from D. The choice of co-occurrence of index terms to define tolerance relation is motivated by its meaningful interpretation of the semantic relation in context of IR and its relatively simple and efficient computation.

Let $D = \{d_1, \ldots, d_N\}$ be a set of documents and $T = \{t_1, \ldots, t_M\}$ set of *index terms* for D. With the adoption of Vector Space Model [1], each document d_i is represented by a weight vector $[w_{i1}, \ldots, w_{iM}]$ where w_{ij} denoted the weight of term t_j in document d_i. TRSM is an approximation space $\mathcal{R} = (T, I_\theta, \nu, P)$ determined over the set of terms T as follows:

– **Uncertain function:** The parameterized uncertainty function I_θ is defined as

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

where $f_D(t_i, t_j)$ denotes the number of documents in D that contain both terms t_i and t_j. Clearly, the above function satisfies conditions of being reflexive: $t_i \in I_\theta(t_i)$ and symmetric: $t_j \in I_\theta(t_i) \iff t_i \in I_\theta(t_j)$ for any $t_i, t_j \in T$. Thus, the tolerance

relation $\mathcal{I} \subseteq T \times T$ can be defined by means of function I_θ:

$$t_i \mathcal{I} t_j \iff t_j \in I_\theta(t_i)$$

where θ is a parameter set by an expert.

The set $I_\theta(t_i)$ is called the *tolerance class* of index term t_i.

– **Vague inclusion function:** To measure degree of inclusion of one set in another, the vague inclusion function is defined as is defined as

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

It is clear that this function is monotonous with respect to the second argument.

– **Structural function:** All tolerance classes of terms are considered as structural subsets: $P(I_\theta(t_i)) = 1$ for all $t_i \in T$.

The membership function μ for $t_i \in T$, $X \subseteq T$ is then defined as $\mu(t_i, X) = \nu(I_\theta(t_i), X) = \frac{|I_\theta(t_i) \cap X|}{|I_\theta(t_i)|}$ and the lower and upper approximations of any subset $X \subseteq T$ can be determined – with the obtained tolerance $\mathcal{R} = (T, I, \nu, P)$ – in the standard way

$$\mathbf{L}_\mathcal{R}(X) = \{t_i \in T \mid \nu(I_\theta(t_i), X) = 1\}$$
$$\mathbf{U}_\mathcal{R}(X) = \{t_i \in T \mid \nu(I_\theta(t_i), X) > 0\}$$

2.3 Example

Consider an universe of unique terms extracted from a set of search result snippets returned from Google search engine for a "famous" query: **jaguar** which is frequently used as a test query in information retrieval because it is a polysemy, i.e., word that has several meanings, especially in the web. The word jaguar can have the following meanings:

– jaguar as a cat (*panthera onca* - http://dspace.dial.pipex.com/agarman/jaguar.htm);

– jaguar as an Jaguar car;

– jaguar was a name for a game console made by Atari - http://www.atari-jaguar64.de,

– it is also a codename for Apple's newest operating system MacOS X - http://www.apple.com/macosx.

Tolerance classes are generated for threshold $\theta = 9$. It is interesting to observe (Table 1) that the generated classes do reveal different meanings of the word "jaguar": a cat, a car, a game console, an operating system and some more.

Term	Tolerance class	Document frequency
Atari	**Atari, Jaguar**	**10**
Mac	**Mac, Jaguar, OS, X**	**12**
onca	onca, Jaguar, Panthera	9
Jaguar	Atari, Mac, onca, Jaguar, club, Panthera, new, information, OS, site, Welcome, X, Cars	185
club	Jaguar, club	27
Panthera	**onca, Jaguar, Panthera**	**9**
new	Jaguar, new	29
information	Jaguar, information	9
OS	Mac,Jaguar, OS, X	15
site	Jaguar, site	19
Welcome	Jaguar, Welcome	21
X	Mac, Jaguar, OS, X	14
Cars	**Jaguar, Cars**	**24**

Table 1. Tolerance classes of terms generated from 200 snippets return by Google search engine for a query "jaguar" with $\theta = 9$;

In context of Information Retrieval, a tolerance class represents a concept that is characterized by terms it contains. By varying the threshold θ , one can control the degree of relatedness of words in tolerance classes (or the preciseness of the concept represented by a tolerance class).

3 Applications of TRS Model in Text Mining

One interpretations of the given approximations can be as follows: if we treat X as an concept described vaguely by index terms it contains, then $\mathbf{U}_{\mathcal{R}}(X)$ is the set of concepts that share some semantic meanings with X, while $\mathbf{L}_{\mathcal{R}}(X)$ is a "core" concept of X. Let us mention two basic applications of TRSM in text mining area:

3.1 Enriching document representation:

In standard Vector Space Model, a document is viewed as a *bag of words/terms*. This is articulated by assigning, non-zero weight values, in document's vector, to terms that occurs in document. With TRSM, the aim is enrich representation of document by taking into consideration not only terms actually occurring document but also other related terms with similar meanings. A "richer" representation of document can be acquired by representing document as set of tolerance classes of terms it contains. This is achieved by simply representing document with its upper approximation, i.e. the document $d_i \in D$ is represented by

$$d_i \dashrightarrow \mathbf{U}_{\mathcal{R}}(d_i) = \{t_i \in T \mid \nu(I_\theta(t_i), d_i) > 0\}$$

In fact, one can apply the enriched representation scheme to any collection of words, e.g., clusters. Moreover, composition the upper approximation operator several times can return even more richer representation of a document, i.e.,

$$d_i \dashrightarrow \mathbf{U}^k_{\mathcal{R}}(d_i) = \underbrace{\mathbf{U}_{\mathcal{R}}(...\mathbf{U}_{\mathcal{R}}}_{\text{k times}}(d_i))$$

The use of upper approximation in similarity calculation to reduce the number of zero-valued similarities is the main advantage the TRSM-based algorithms claimed to have over traditional approaches. This makes the situation, in which two documents have a non-zero similarity although they do not share any terms, possible.

3.2 Extended weighting scheme:

To assign weight values for document's vector, the TF*IDF weighting scheme is used. In order to employ approximations for document, the weighting scheme need to be extended to handle terms that occurs in document's upper approximation but not in the document itself. The extended weighting scheme is defined from the standard TF*IDF by:

$$w_{ij}^{*} = \begin{cases} (1 + \log f_{d_i}(t_j)) \log \frac{N}{f_D(t_j)} & \text{if } t_j \in d_i \\ 0 & \text{if } t_j \notin \mathbf{U}_{\mathcal{R}}(d_i) \\ \min_{t_k \in d_i} w_{ik} \frac{\log \frac{N}{f_D(t_j)}}{1+\log \frac{N}{f_D(t_j)}} & \text{otherwise} \end{cases}$$

The extension ensures that each terms occurring in upper approximation of d_i but not in d_i, has a weight smaller than the weight of any terms in d_i. Normalization by vector's length is then applied to all document vectors:

$$w_{ij}^{new} = \frac{w_{ij}^{*}}{\sqrt{\sum_{t_k \in d_i} (w_{ij}^{*})^2}}$$

4 Document clustering algorithms based on TRSM

With the introduction of Tolerance Rough Set Model in [16], several document clustering algorithms based on that model was also introduced [16, 19]. The main novelty that TRSM brings into clustering algorithms are the way of representing clusters and document.

4.1 Cluster representation:

Determining cluster representation is very important factor in partitioning-based clustering. Frequently, cluster is represented as a mean or median of all documents it contains. Sometime however, a representation not based on vector is needed as

Title: EconPapers: Rough sets bankruptcy prediction models versus auditor
Description: Rough sets bankruptcy prediction models versus auditor signalling rates. Journal of Forecasting, 2003, vol. 22, issue 8, pages 569-586. Thomas E. McKee. ...

Original vector		**Enriched vector**	
Term	Weight	Term	Weight
auditor	0.567	auditor	0.564
bankruptcy	0.4218	bankruptcy	0.4196
signalling	0.2835	signalling	0.282
EconPapers	0.2835	EconPapers	0.282
rates	0.2835	rates	0.282
versus	0.223	versus	0.2218
issue	0.223	issue	0.2218
Journal	0.223	Journal	0.2218
MODEL	0.223	MODEL	0.2218
prediction	0.1772	prediction	0.1762
Vol	0.1709	Vol	0.1699
		applications	0.0809
		Computing	0.0643

Table 2. Example of snippet and its two vector representations.

cluster description is directly derived from its representation. For example, cluster can be represented by most "distinctive" terms from cluster's documents (e.g. most frequent terms; or frequent terms in clusters but infrequent globally).

In [19], an approach to construct a *polythetic* representation is presented. Let R_k denotes a representative for cluster k. The aim is to construct a set of index terms R_k representing cluster C_k so that:

- each document d_i in C_k share some or many terms with R_k
- terms in R_k occurs in most documents in C_k
- terms in R_k needs not to be contained by every document in C_k

The weighting for terms t_j in R_k is calculated as an averaged weight of all occurrences in documents of C_k:

$$w_{kj} = \frac{\sum_{d_i \in C_k} w_{ij}}{|\{d_i \in C_k \mid t_j \in d_i\}|}$$

Let $f_{C_k}(t_J)$ be the number of documents in C_k that contain t_j. The above assumptions lead to following rules to create cluster representatives:

Algorithm 1 Determine cluster representatives

1: $R_k = \emptyset$
2: **for all** $d_i \in C_k$ and $t_j \in d_i$ **do**
3: **if** $f_{C_k}(t_j)/|C_k| > \sigma$ **then**
4: $R_k = R_k \cup t_j$
5: **end if**
6: **end for**
7: **if** $d_i \in C_k$ and $d_i \cap R_k = \emptyset$ **then**
8: $R_k = R_k \cup \text{argmax}_{t_j \in d_i} w_{ij}$
9: **end if**

To the initially empty representative set, terms that occur frequent enough (controlled by threshold σ) in documents within cluster are added. After this phase, for each document that is not yet "represented" in representative set (i.e. document shares no terms with R_k), the strongest/heaviest term from that document is added to the cluster representatives.

4.2 TRSM-based clustering algorithms:

Both hierarchical and non-hierarchical clustering algorithms were proposed and evaluated with standard test collections and have shown some successes [16, 19].

The non-hierarchical document clustering algorithm based on TRSM is a variation of a K-means clustering algorithm with overlapping clusters. The modifications introduced are:

- use of document's upper approximation when calculating document-cluster and document-document similarity,
- documents are soft-assigned to cluster with associated membership value
- use nearest-neighbor to assign unclassified documents to cluster

A hierarchical agglomerative clustering algorithm based on TRSM utilizes upper approximation to calculate cluster similarities in the merging step.

5 Case Study: The TRSM-based Snippets Clustering Algorithm

The Tolerance Rough set Clustering algorithm is based primarily on the K-means algorithm presented in [19]. By adapting K-means clustering method, the algorithm remain relatively quick (which is essential for on-line results post-processing) while still maintaining good clusters quality. The usage of Tolerance Space and upper approximation to enrich inter-document and document-cluster relation allows the algorithm to discover subtle similarities not detected otherwise. As it has been mentioned, in search results clustering, the proper labelling of cluster is as important as cluster contents quality. Since the use of phrases in cluster label has been proven [23, 8] to be more effective than single words, TRC algorithm utilize n-gram of words (phrases) retrieved from documents inside cluster as candidates for cluster description.

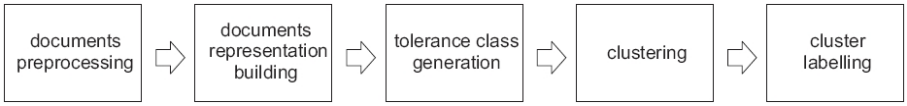


Fig. 1. Phases of TRC algorithm

The TRC algorithm is composed from five phases (depicted in Fig .1):

1. **documents preprocessing:** In TRC, the following standard preprocessing steps are performed on snippets: *text cleansing, text stemming, and Stop-words elimination.*
2. **documents representation building:** In this step, two main procedures are performed: *index term selection and term weighting.*

3. **tolerance class generation:** The goal of the tolerance class generation is to determine for each term, set of its related terms with regards to the tolerance relation – the tolerance class.
4. **clustering:** Applying k-means clustering algorithm based on TRSM.
5. **cluster labelling**: For labelling cluster we have decided to employs phrases because of its high descriptive power [23, 8]. We have adapted an algorithm for n-gram generation from [18] to extract phrases from contents of each cluster. Most descriptive phrases are chosen to serve as labels for cluster.

The details of the last three steps are described in the following Sections.

While document clustering deals with full-size documents, in clustering search results we have only set of small-size snippets.

5.1 Preprocessing

It is widely known [4] that preprocessing text data before feeding it into clustering algorithm is essentials and can have great impact on algorithm performance. In TRC, several preprocessing steps are performed on snippets:

Text cleaning In this step, text content of snippet is cleaned from unusable terms such as:

- non-letter characters like , $, #
- HTML-related tags (e.g.= `<a>`, `<p>`) and entities (e.g. &amp, &quot)

Stemming A version of Porter's stemmer [14] is used in this step to remove prefixes and suffixes, normalizing terms to its root form. This process can greatly reduce vocabulary of the collection without much semantic loss. The stemmed terms are linked to its original form, which are preserved to be used in subsequent phases (i.e. labels generation).

Stop-words elimination A stop-word itself doesn't bring any semantic meanings but in connection with other words can form meaningful phrases. Therefore, terms that occur in stop-word list are specially marked to be ignored from document index terms, but not removed (so it can uses in phrases extraction in label generation phase). Due to special nature of web document, some words like "web", "http", "site" appear very frequently, thus a stop-word list adapted to web vocabulary from [21] is used.

5.2 Document corpus building

As TRC utilizes Vector Space Model for creating document-term matrix representing documents.

Index terms selection Index terms are selected from all unique stemmed terms after stop-words elimination and with regards to the following rules:

- digits and terms shorter than 2 characters are ignored
- terms contained in the query are ignored (as we are operating on the top search results for a query, terms from query will occurs in almost every snippet)
- Minimum Document Frequency filtering - term that occurs in less given threshold (e..g less than 2 snippets) are ignored as they will doubly contributes to document characterization

Selected index terms used to characterize documents are enumerated and a document-term frequency matrix is built. Let N be a number of document – search results snippets, and M is the number of selected index terms. The document-term frequency matrix is defined as follows:

$$TF = [tf_{i,j}]_{N \times M}$$

where $tf_{i,j}$ — number of occurrences of term j in document j. Each row $TF[i]$ of TF matrix is a characterization of the i-th document by means of term frequencies.

Term weighting The TF*IDF term weighting scheme is applied for document-term matrix to create document-term weight matrix:

$$W = [w_{i,j}]_{N \times M}$$

where $w_{i,j}$ — weight of term j in i-th document. Each row $W[i]$ in the W matrix represents a characterization of the i-th document by means of weighted terms.

5.3 Tolerance class generation

This phase exists for the computational optimization purpose. It ensures that the calculation of the upper approximation for a set of terms can be done quickly.

Let us define *term co-occurrence matrix* as

$$TC = [tc_{x,y}]_{M \times M}$$

where $tc_{x,y}$ is a *co-occurrence frequency* of two terms x, y — number of documents in the collection in which terms x and y both occur. Let tolerance relation R between terms be defined as

$$xRy \iff tc_{x,y} > \theta$$

where θ is called **co-occurrence threshold**. Having term co-occurrence matrix calculated we can define tolerance relation with different granularity by varying co-occurrence threshold.

Figure 2 presents the main steps of the tolerance class generation phase. The computation cost of step 1 is $O(N \times M)$, step 2 and 3 are both $O(M^2)$, altogether is $O(N \times M^2)$.

The detail implementation of this phase is presented in Algorithm 5.3.

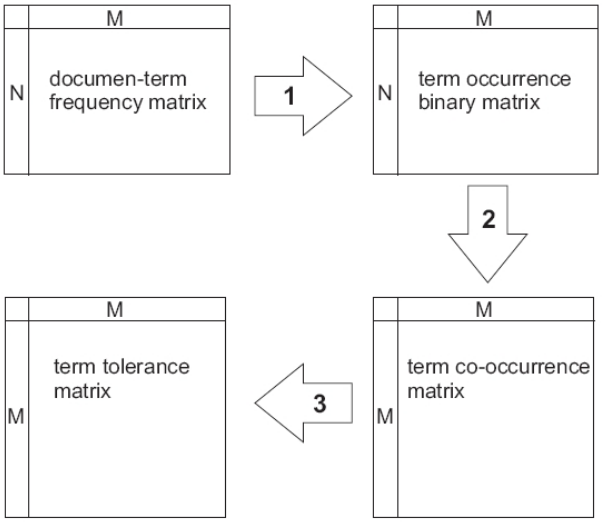


Fig. 2. Process of generating tolerance classes

5.4 K-means clustering

TRC adapts a variation of K-means algorithm for creating groups of similar snippets. Several main steps of the algorithm are described below (see pseudo-code in Algorithm 3):

Initial cluster forming: Selected documents serve as initial cluster representatives

$R_1, R_2, ..., R_K$.

Stop condition: The fact that the clustering algorithm is used as a post-retrieval[4] process puts a strict constraint on execution time, as users are not willing to wait more than a few seconds for a response. We decided to set a limit of **maximum number of iteration** for the K-means algorithm. Due to the nature of quick convergence of K-means algorithm, this limit allows us to reduce the response time of the clustering engine with insignificant lost in clustering quality.

[4] Results returned from search engine is processed on-the-fly by the clustering algorithm and presented to the user

Algorithm 2 Tolerance class generation (Figure 2)

Input: TF – document-term frequency matrix, θ – co-occurrence threshold

Output: TOL – term tolerance binary matrix defining tolerance classes of term

1: Calculate a binary occurrence matrix OC based on document-term frequency matrix TF as follows: $OC = [oc_{i,j}]_{N \times M}$ where

$$oc_{i,j} = \begin{cases} 1 \text{ if } tf_{i,j} > 0 \\ 0 \text{ otherwise} \end{cases}$$

Each column in OC is a bit vector representing term occurrence pattern in a document — bit is set if term occurs in a document.

2: Construct term co-occurrence matrix $COC = [coc_{x,y}]_{M \times M}$ as follows: for each pair of term x, y represented as pair of columns $OC[x], OC[y]$ – bit vectors – in the OC matrix

$$coc_{x,y} = \text{card}(OC^x \text{AND} OC^y)$$

where AND is a binary AND between bit vectors and *card* return cardinality – number of bits set – of a bit vector. $boc_{x,y}$ is the co-occurrence frequency of term x and y[3].

3: Given a co-occurrence threshold θ, a term tolerance binary matrix $TOL = [tol_{x,y}]_{M \times M}]$ can be easily constructed by filtering out cells with values smaller than threshold θ :

$$tol_{x,y} = \begin{cases} 1 \text{ if } coc_{x,y} \geq \theta \\ 0 \text{ othewise} \end{cases}$$

Each row in the resulting matrix forms a bit vector defining a tolerance class for given term: $tol_{x,y}$ is set if term x and y are in tolerance relation.

Algorithm 3 Clustering phase of TRC algorithm

Input: D – set of N documents, K – number of clusters, δ – cluster similarity threshold

Output: K overlapping clusters of documents from D with associated membership value

```
1: Randomly select K document from D to serve as K initial cluster C_1, C_2, ..., C_K.
2: repeat
3:   for each d_i ∈ D do
4:     for each cluster C_k, k = 1, .., K do
5:       calculate the similarity between document's upper approximation and the
         cluster representatives S(U_R(d_i), R_k)
6:       if S(U_R(d_i), R_k) > δ then
7:         assign d_i to C_k with the similarity taken as cluster membership:
           m(d_i, C_k) = S(U_R(d_i), R_k)
8:       end if
9:     end for
10:  end for
11:  for each cluster C_k do
12:    determine_cluster_representatives(R_k)
13:  end for
14: until stop_condition()
15: post-process unassigned documents
16: if necessary determine_cluster_representatives(R_k) for changed clusters
    C_k
```

Determining cluster representatives: Cluster representatives is determined as described in Section 4.1

Nearest neighbor assignment: As a result of the restriction set by **cluster similarity threshold**, after all iterations there may exist document that has not been assigned to any cluster. In TRC, there are two possible options:

- create a special "Others" cluster with unassigned documents as proposed by [8]
- assigns these documents to their nearest cluster

For the later options, we have decided to assign document to its nearest neighbor's cluster.

Cluster label generation: As already pointed out, when evaluating clustering algorithms for search results, the quality of cluster label is as much important

as the quality of the cluster itself. For labelling cluster we have decided to employs phrases because of its high descriptive power [23, 8]. We have adapted an algorithm for n-gram generation from [18] to extract phrases from contents of each cluster. Most descriptive phrases are chosen to serve as labels for cluster. Descriptiveness of phrases is evaluated by taking into consideration following criteria:

- frequency of the phrase in the whole collection
- frequency of the phrase inside a cluster
- length of the phrase, measured as number of words it is made from

Following the intuition of TD*IDF scheme, we hypothesize that phrases that are relatively infrequent in the whole collection but occurs frequently in clusters will be good candidate for cluster's label. We also prefer long phrases over shorter one.

Algorithm 4 Assignment of document to its nearest neighbor's cluster

for each unassigned document d_u **do**
  Find the nearest neighbor document $NN(d_u)$ with non-zero similarity;
  Amongst clusters which $NN(d_u)$ belongs to; choose the one C_k that $NN(d_u)$ has strongest membership with;
  Assign d_u to C_k and calculate its cluster membership as $m(d_u, C_k) = m(NN(d_u), C_k) \cdot S(\mathbf{U}_{\mathrm{R}}(d_u), \mathbf{U}_{\mathrm{R}}(NN(d_u)))$;
end for

6 Experimental Results

Due to aforementioned lack of standard collection for testing web search results clustering, we had to build a small test collection. For this purpose, we have defined a set of queries for which results were collected from major search engine Google to form test data collection.

6.1 Test queries

The test set is composed from queries representing subjects of various degree of specificity, in order to test algorithm behavior on data with different vocabulary characteristic.

Query	Results count	Specificity	Snippets	Terms	Terms per snippet Ave.	Min	Max
java	23400000	low	200	332	7.280	0	15
clinton	4270000	low	200	337	7.305	0	13
jaguar	2580000	low	200	339	6.595	0	13
"data mining"	1080000	medium	200	319	8.815	0	16
wifi	864000	medium	200	340	6.915	1	19
clustering	810000	medium	195	319	7.456	0	16
voip	916000	high	200	355	8.235	0	17
"rough sets"	748	high	146	244	9.089	1	18
"search results clustering"	105	high	68	149	12.279	2	19

Table 3. Queries used to generate test collection and characteristics of retrieved snippets from Google for those queries.

The first three queries represent very general concepts and are frequently used as a test by authors [23, 20] of search results clustering algorithm. Next three queries are more specific subjects but broad enough to have interesting subclasses. Last three queries are about relatively specific topics and were chosen to test algorithm on highly cohesive vocabulary. In Table 3, figures in the second column are the *approximated number of results* for a queries retrieved from Google. Looking at these figures gives us some additional intuition about generality (or specificity) of concepts represented by query, or just popularity of each subject on the web.

Table 3 shows some statistics of snippet collection retrieved from Google search engine for set of test queries. Both stemming and stop-word list were used to filter out unnecessary terms. Additionally Minimum Document Frequency filter was used to

remove terms that occurs in less than 2 documents. Thus, the indexing vocabulary could be reduced by about 70%. On average, any snippet is indexed by 6 to 10 terms, which is about 2-3% of total vocabulary. Worth noticed in the results of term filtering, some document may be left represented by none terms.

6.2 Inter-document similarity enrichment

The main purpose of using upper approximation in our TRC algorithm is to enhance the association between documents, i.e., to increase the similarity between related documents. To measure that enhancement, we compare the *density* of similarity functions created by standard document representation and the one using upper approximation (for all collections of snippets). In our experiments, the similarity between two documents d_i and d_j , also called *inter-document similarity*, is calculated by using cosine similarity measure (see [15]), and denoted by $sim(d_i, d_j)$. The density of a given similarity function $sim : D \times D \rightarrow [0, 1]$ over a collection D of documents is calculated as a number of pairs $(d_i, d_j) \in D \times D$ of documents such that $sim(d_i, d_j) > t$, where $t \in [0, 1]$ is called *a similarity level*.

For a given collection D of snippets and the similarity level t, the *relative density improvement of similarity function*, denoted by $improve_t(D)$, is measured by

$$\frac{dense_t(sim_{TRSM}) - dense_t(sim)}{dense_t(sim)}$$

where $dense_t(sim)$ and $dense_t(sim_{TRSM})$ are densities of two similarity functions defined by two document representations: the standard and the one based on TRSM. In Figure 3 (up), the *relative density improvement* of similarity function for tested snippet collections in different similarity levels are presented.

It can be observed that the enrichment of upper approximation had indeed improved inter-document similarities for all queries. The level of improvement varies with different queries depending on the co-occurrence threshold. Some queries like

"java", "clinton", "voip", achieved significantly better level of improvement than others ("jaguar", "clustering"). It is promising that improvement has been happened in all similarity levels (the improvement in level $t = 0.67$ were significantly improved). This is very important for clustering quality as this could forms better clusters. The representation enrichment technique results in improvement of the clustering process as it is presented in Figure 3 (bottom).

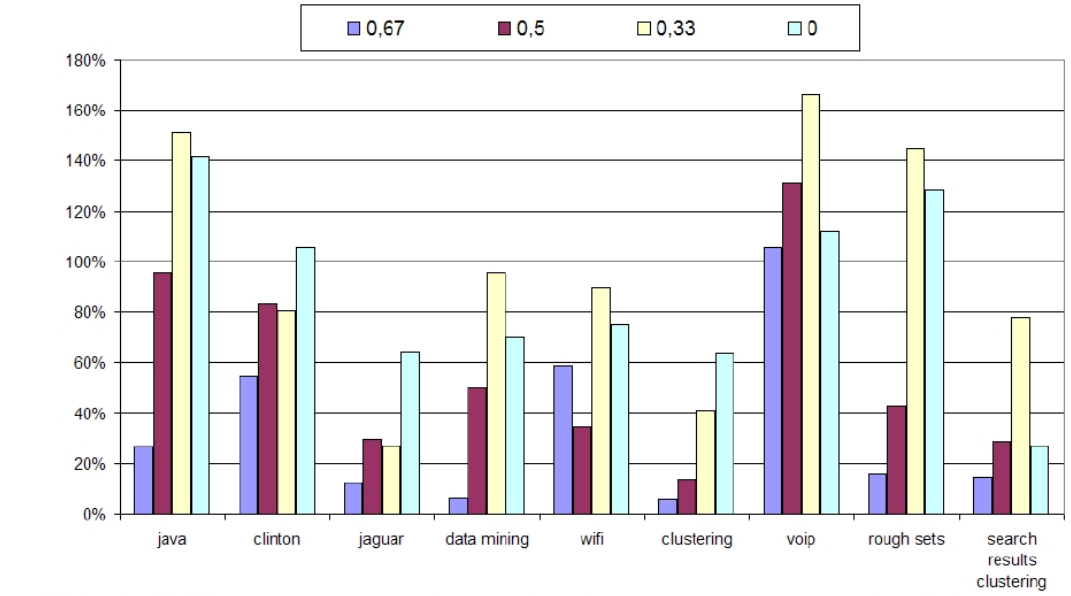
6.3 Comparison with other approaches

The TRC algorithm was implemented entirely in Java programming language, as a component within the $Carrot^2$ framework [6]. $Carrot^2$ is an open-source, data processing framework developed by Weiss [9]. It enables and ease experimentation of processing and visualization of search results. The $Carrot^2$ architecture is based on a set of loosely-coupled but cooperating components that exchanges information with each other using XML messages.

Such implementation of TRC makes the comparison of TRC with other algorithms like LINGO [8], AHC [22], STC [20] possible, because all algorithms (included our own TRC) were developed within the *Carrot2* Framework [9]. This makes possible to compare different algorithms running in the same environment.

7 Conclusions

This paper was thought as a proof-of-concept demonstration of Tolerance Rough Set application to web mining domain. We wanted to investigate how rough set theory and its ideas, like approximations of concept, could be practically applied in a task of search results clustering. The result is a design of a TRC — a Tolerance Rough Set Clustering algorithm for web search results and implementation of the proposed solution within an open-source framework, $Carrot^2$.



sub topics

All groups (133)
- Jaguar Drivers Club (10)
- Jaguar Drivers Club (4)
- Club (16)
- Announces The Release (2)
- Bumper (3)
- Jaguar Cars (9)
- Auto (11)
- Apple Previews Jaguar (8)
- Wildlife Conservation Society (2)
- Jaguar Technologies (3)
- Panthera Onca 35k (5)
- Parts (7)
- Friendly Page Send (2)
- Jaguar (5)
- Other (46)

sub topics

All groups (138)
- Jaguar Drivers Club (10)
- Auto (11)
- Jaguar Panthera Onca (6)
- Jaguar Racing (2)
- Jaguar Drivers Club (4)
- Parts (9)
- Announces The Release (2)
- Server With Jaguar (4)
- British Rock Band (4)
- Jaguar Cars (16)
- Jaguar Austria (2)
- Website Van Jaguar (2)
- Jaguar Technologies (3)
- Panthera Onca 35k (5)
- Classic Jaguar (5)
- Jaguar Owners (3)
- Apple Previews Jaguar (6)
- Atari Jaguar Faq (5)
- Jaguar Cars (9)
- Jaguar Films (3)
- Other (27)

sub topics

All groups (144)
- Jaguar Drivers Club (10)
- Apple Previews Jaguar (6)
- Auto (12)
- Jaguar Parts (8)
- Atari Jaguar Faq (4)
- Bumper (3)
- Jaguar Racing (2)
- Jaguar Films (3)
- Club (16)
- Announces The Release (2)
- Server With Jaguar (4)
- Quark (3)
- Jaguar Owners (3)
- Wildlife Conservation Society (2)
- Jaguar Austria (2)
- Jaguar Association Germany (2)
- Website Van Jaguar (2)
- Jaguar Technologies (3)
- Endangered Description (1)
- Sportsbook (2)
- Panthera Onca 35k (5)
- Classic Jaguar (5)
- Friendly Page Send (2)
- Jaguar Cars (9)
- Jaguar Panthera Onca (6)
- Atari Jaguar Faq (5)
- Other (22)

Fig. 3. Top: Relative improvement of inter-document similarity matrix measure with co-occurrence threshold = 5 and various similarity levels $t = 0, 0.33, 0.5, 0.67$. Bottom: Example of clustering results produced by TRC for query "jaguar" with different similarity thresholds

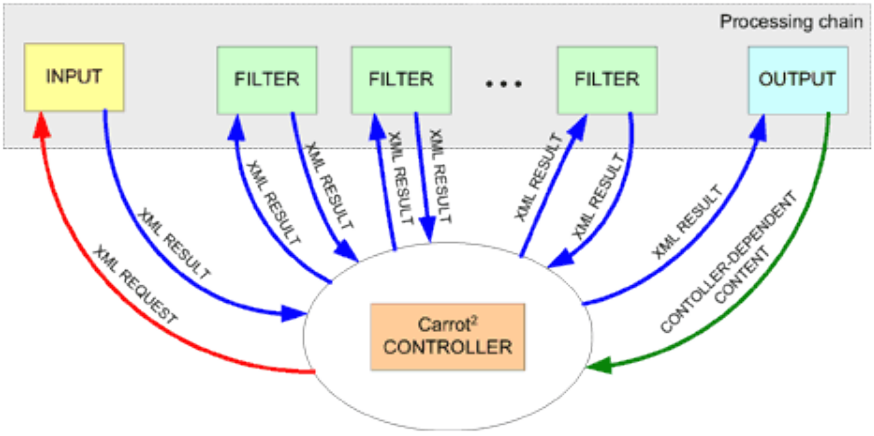


Fig. 4. Data flow in $Carrot^2$ framework

The experiment we have carried has been showed that Tolerance Rough Set and upper approximation it offers can indeed improve the representations, which have positive effects on clustering quality. The results are promising and encourage further work on its application. This research is a step forward to make the application of Computing with Words (CW) in the internet possible.

Acknowledgement:

The research has been partially supported by the grant 3T11C00226 from the Ministry of Scientific Research and Information Technology of the Republic of Poland

References

1. Baeza-Yates, R., and Ribeiro-Neto, B. *Modern Information Retrieval*, 1st ed. Addison Wesley Longman Publishing Co. Inc., May 1999.
2. Bargiela, A., and Pedrycz, W. *Granular Computing. An Introduction.* The International Series in Engineering and Computer Science, Vol. 717. Kluwer Academic Publishers, Boston, MA, USA, 2002.
3. Hearst, M. A., and Pedersen, J. O. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval* (Zürich, CH, 1996), pp. 76–84.
4. Jiawei Han, M. K. *Data Mining: Concepts and Techniques*, 1st ed. Morgan Kaufmann, 2000.
5. Komorowski, J., Pawlak, Z., Polkowski, L., and Skowron, A. Rough sets: a tutorial, 1998.

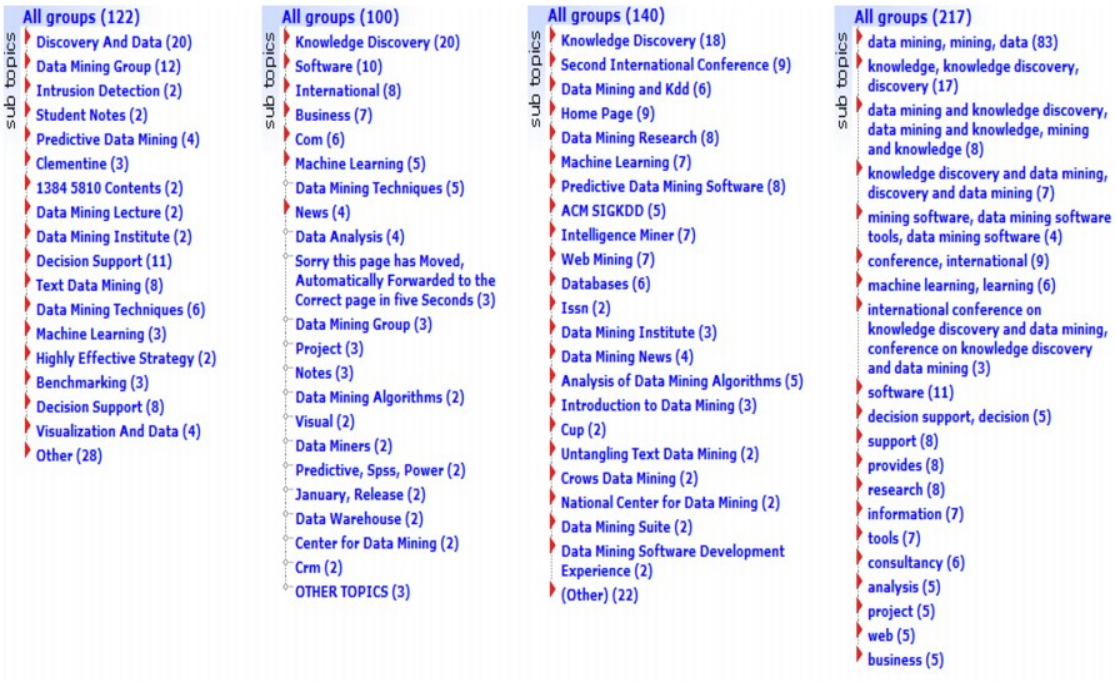


Fig. 5. Comparison of results for a query "data mining" produced by different algorithm (from left to right): TRC, LINGO, AHC and STC. All output were produced using *Carrot*^{2} visualization component.

6. Ngo, C. L., and Nguyen, H. S. A method of web search result clustering based on rough sets. In *Web Intelligence* (2005), A. Skowron, R. Agrawal, M. Luck, T. Yamaguchi, P. Morizet-Mahoudeaux, J. Liu, and N. Zhong, Eds., IEEE Computer Society, pp. 673–679.
7. Nguyen, H. S., Skowron, A., and Stepaniuk, J. Granular computing: a rough set approach. *Computational Intelligence: An International Journal 17*, (no. 3) (August 2001), 514–544(31.
8. Osinski, S. An algorithm for clustering of web search result. Master's thesis, Poznan University of Technology, Poland, June 2003.
9. Osinski, S., and Weiss, D. Carrot^2: Design of a flexible and efficient web information retrieval framework. In *AWIC* (2005), P. S. Szczepaniak, J. Kacprzyk, and A. Niewiadomski, Eds., vol. 3528 of *Lecture Notes in Computer Science*, Springer, pp. 439–444.
10. Osiński, S., and Weiss, D. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems 20*, 3 (2005), 48–54.
11. Pawlak, Z. *Rough sets: Theoretical aspects of reasoning about data.* Kluwer Dordrecht, 1991.
12. Pawlak, Z. Granularity of knowledge, indiscernibility, and rough sets. In *Proceedings: IEEE Transactions on Automatic Control 20* (1999), pp. 100–103.
13. Polkowski, L. T., and Skowron, A. Towards adaptive calculus of granules. In *Proceedings of the FUZZ-IEEE International Conference, 1998 IEEE World Congress on Computational Intelligence (WCCI'98)* (1998), pp. 111–116.

14. Porter, M. F. An algorithm for suffix stripping. In *Readings in Information Retrieval*, P. W. Karen Sparck Jones, Ed. Morgan Kaufmann, San Francisco, 1997, pp. 130–137.
15. Salton, G. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., 1989.
16. Saori Kawasaki, Ngoc Binh Nguyen, T. B. H. Hierarchical document clustering based on tolerance rough set model. In *Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000, Lyon, France, September 13-16, 2000, Proceedings* (2000), D. A. Zighed, H. J. Komorowski, and J. M. Zytkow, Eds., vol. 1910 of *Lecture Notes in Computer Science*, Springer.
17. Skowron, A., and Stepaniuk, J. Tolerance approximation spaces. *Fundamenta Informaticae 27*, 2-3 (1996), 245–253.
18. Smadja, F. A. From n-grams to collocations: An evaluation of xtract. In *29th Annual Meeting of the Association for Computational Linguistics, 18-21 June 1991, University of California, Berkeley, California, USA, Proceedings* (1991), pp. 279–284.
19. Tu Bao Ho, N. B. N. Nonhierarchical document clustering based on a tolerance rough set model. *International Journal of Intelligent Systems 17*, 2 (2002), 199–212.
20. Weiss, D. A clustering interface for web search results in polish and english. Master's thesis, Poznan University of Technology, Poland, June 2001.
21. Weiss, D. A clustering interface for web search results in polish and english, 2001.
22. Wroblewski, M. A hierarchical www pages clustering algorithm based on the vector space model. Master's thesis, Poznan University of Technology, Poland, July 2003.
23. Zamir, O., and Etzioni, O. Grouper: a dynamic clustering interface to web search results. *Computer Networks (Amsterdam, Netherlands: 1999) 31*, 11–16 (1999), 1361–1374.
24. Asharaf S, Shevade SK, Murty NM (2005) Rough Support Vector Clustering, Pattern Recognition 38(10):1779-1783
25. Hirano S and Tsumoto S (2000) Rough Clustering and Its Application to Medicine. Journal of Information Science 124:125-137
26. Lingras P (2001) Unsupervised Rough Set Classification using GAs. Journal Of Intelligent Information Systems 16(3):215-228.
27. Lingras P, West C (2004) Interval Set Clustering of Web Users with Rough K-means. Journal of Intelligent Information Systems 23(1):5-16
28. Lingras P, Hogo M, Snorek M (2004) Interval Set Clustering of Web Users using Modified Kohonen Self-Organizing Maps based on the Properties of Rough Sets. Web Intelligence and Agent Systems: An International Journal 2(3):217-230.
29. Peters JF, Skowron A, Suraj Z, Rzasa W, Borkowski M (2002) Clustering: A rough set approach to constructing information granules. Soft Computing and Distributed Processing, Proceedings of 6th International Conference, SCDP 2002, 57-61

30. Voges KE, Pope NKLl, Brown MR (2002) Cluster Analysis of Marketing Data: A Comparison of K-Means, Rough Set, and Rough Genetic Approaches. In: Abbas HA, Sarker RA, Newton CS (eds) Heuristics and Optimization for Knowledge Discovery, Idea Group Publishing 208-216.