A method of web search result clustering based on rough sets

Chi Lang Ngo, Hung Son Nguyen (*)
Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
(*) E-mail (contact person): son@mimuw.edu.pl

Abstract

Due to the enormous size of the web and low precision of user queries, finding the right information from the web can be difficult if not impossible. One approach that tries to solve this problem is using clustering techniques for grouping similar document together in order to facilitate presentation of results in more compact form and enable thematic browsing of the results set. The main problem of many web search result (snippet) clustering algorithm is based on the poor vector representation of snippets. In this paper, we present a method of snippet representation enrichment using Tolerance Rough Set model. We applied the proposed method to construct a rough set based search result clustering algorithm and compared it with other recent methods.
Keywords: *rough sets, snippet, clustering.*

1. Introduction

Two most popular approaches to facilitate searching for information on the web are represented by web search engine and web directories. Web search engines 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 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 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.

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). 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.

The earliest work on clustering results were done by Pedersen, Hearst et al. on Scather/Gather system [2], followed with application to web documents and search results by Zamir et al. [15] to create Grouper based on novel algorithm Suffix Tree Clustering. Inspired by their work, a Carrot framework was created by Weiss [12] to facilitate research on clustering search results. This has encouraged others to contribute new clustering algorithms under the Carrot framework.

In this paper, we proposed an approach to search results clustering based on Tolerance Rough Set following the work on document clustering of Bao [4, 3]. 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. Tolerance Rough Set Model

Rough set theory was originally developed [8] 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.

Consider a non-empty set of object U called the universe. Suppose we want to define a concept over universe of objects 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*.

The classical rough set theory is based on equivalence relation that divides the universe of objects into disjoint classes. For some practical 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. 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 [10] 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,

$I : U \to \mathcal{P}(U)$, 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,

$\nu : \mathcal{P}(U) \times \mathcal{P}(U) \to [0, 1]$ 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 $X, Y, Z \subseteq U$.

$P : I(U) \to \{0, 1\}$ 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.

Tolerance Rough Set Model (TRSM) was developed [4, 3] 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 [3] that a single document is usually represented by relatively few terms. 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 parameterize 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. The set $I_\theta(t_i)$ is called the *tolerance class* of index term t_i.

Vague inclusion function: is defined as

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

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

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.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 $\theta = 9$;

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

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

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). 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:

2.1. 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

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

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.

2.2. 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:

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

The use of upper approximation in similarity calculation to reduce the number of zero-valued similarities is the main advantage main advantage 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. The TRC algorithm

Any clustering algorithm can be applied to the extended representation scheme of snippets. We have implemented the Tolerance Rough set Clustering (TRC) algorithm which is based primarily on the K-means algorithm presented in [3]. By adapting K-means clustering method, the algorithm remain relatively quick 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. The TRC algorithm is composed by following phases:

Phase 1: Document preprocessing: *cleansing, stemming, and stop-words elimination*

Phase 2: Document representation building: *index term selection and term weighting*;

Phase 3: Tolerance class generation;

Phase 4: Clustering: TRC adapts a variation of K-means algorithm for creating groups of similar snippets;

Phase 5: Cluster labelling;

The goal of the tolerance class generation phase is to determine for each term, set of its related terms with regards to the tolerance relation – the tolerance class. Let us define *term co-occurrence matrix* as $TC = [tc_{x,y}]_{M \times M}$, where $tc_{x,y}$ is the 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**. The computation cost of this phase is $O(N \times M^2)$, where N is the number of snippets and M is the number of index terms for the collection.

In TRC algorithm, we apply the approach to construct a *polythetic* representation presented in [3]. The weight v_{kj} of term t_j in R_k is calculated as an averaged weight of all occurrences in documents of C_k, i.e., $v_{kj} = \sum_{d_i \in C_k} w_{ij} / f_{C_k}(t_j)$, where $f_{C_k}(t_j) = |\{d_i \in C_k \mid t_j \in d_i\}|$ is the number of documents in C_k that contain t_j. The terms that occur frequent enough (controlled by threshold σ) in documents within cluster are chosen. To ensure that each document is "represented" in the representative set the strongest term from each document is added to the cluster representatives. The above assumptions lead to following rules to create cluster representatives:

$$R_k = \{t_j : v_{kj} > \sigma\} \cup \bigcup_{d_i \in C_k} \left\{ \text{argmax}_{t_j \in d_i} w_{ij} \right\}$$

For labelling cluster we have decided to employ phrases because of its high descriptive power [15]. We have adapted an algorithm for n-gram generation from [11] to extract phrases from contents of each cluster. Most descriptive phrases are chosen to serve as labels for cluster. Following the intuition of TD*IDF, we hypothesize that phrases that are relatively infrequent in the whole collection but occurs frequently in clusters are good candidates for cluster's label. We also prefer long phrases over shorter one.

4. Experimental Results

Due to aforementioned lacks 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. The first three queries represent very general concepts and are frequently used as a test by authors [15, 12] 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. 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.

4.1. 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

Query	Results count	Specificity	Snippets	Terms	Terms per snippet Average	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.

measure (see [9]), 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 1 (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 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 1 (bottom).

4.2. 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 [13]. 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 [7], AHC [14], STC [12] possible, because all algorithms (included our own TRC) were developed within the *Carrot2* Framework [13]. This has made possible to compare algorithms running in the same environment.

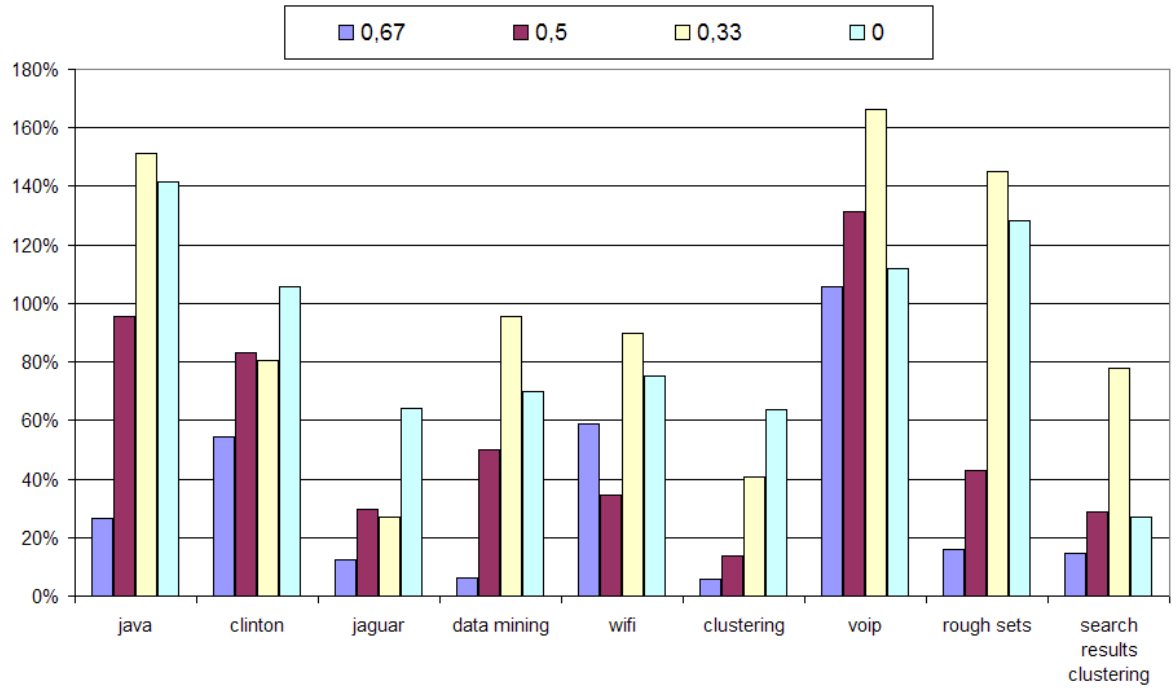
5. Conclusions

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, $Carrot^2$.

The experiment we have carried has 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.

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



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)

Figure 1. 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

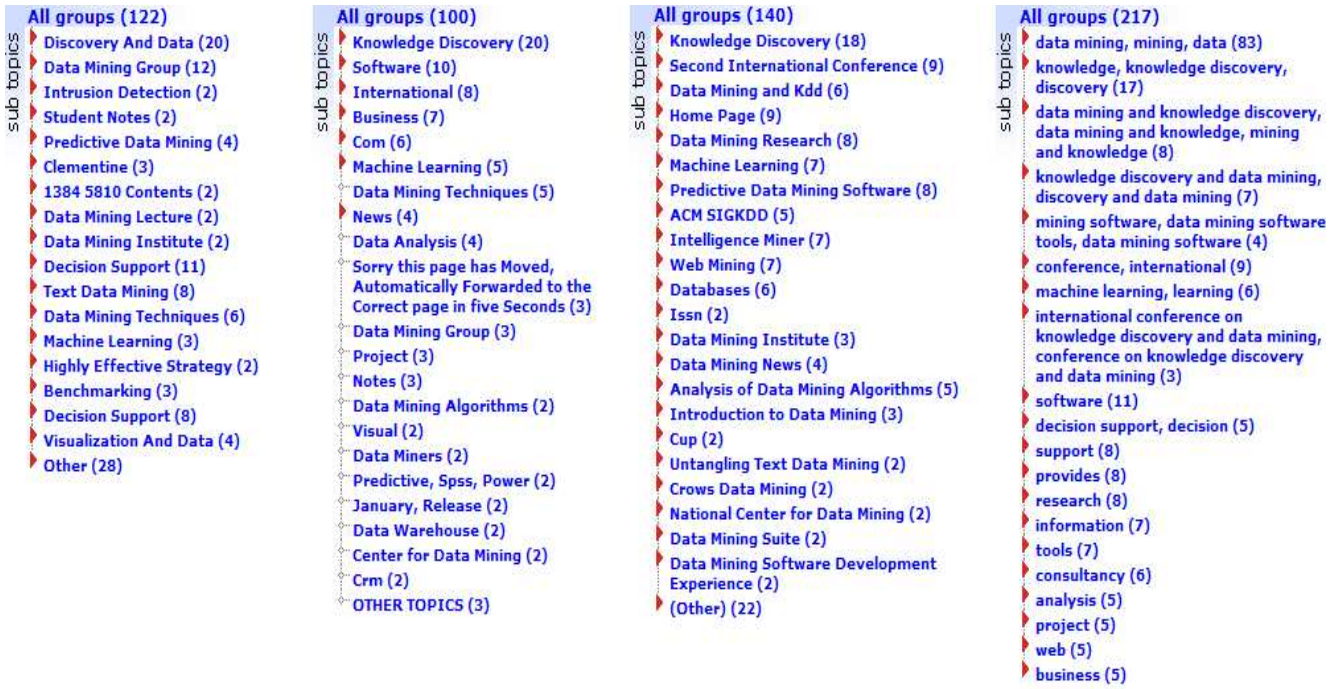


Figure 2. Comparison of results for a query "data mining" produced by different algorithm (from left to right): TRC, LINGO, AHC and STC. All output were produced using $Carrot^2$ visualization component.

References

[1] Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. 1st edn. Addison Wesley Longman Publishing Co. Inc. (1999)

[2] Hearst, M.A., 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), 76–84

[3] Ho, T.B, Nguyen, N.B.: Nonhierarchical document clustering based on a tolerance rough set model. International Journal of Intelligent Systems **17** (2002) 199–212

[4] Ho, T.B., Kawasaki, S., Nguyen, N.B.: Documents clustering using tolerance rough set model and its application to information retrieval. In Intelligent exploration of the web, 2003, Physica-Verlag, 181–196

[5] Komorowski, J., Pawlak, Z., Polkowski, L., Skowron, A.: Rough sets: a tutorial (1998)

[6] Chi Lang Ngo, Hung Son Nguyen: Tolerance rough set approach to clustering web search results, J.-F.Boulicaut, F.Esposito, F.Gianotti, and D.Pedreschi (Eds.): Knowledge Discovery in Databases: Proceedings of PKDD 2004, LNAI 3302, 513–517.

[7] Osinski, S.: An algorithm for clustering of web search result. Master's thesis, Poznan University of Technology, Poland (2003)

[8] Pawlak, Z.: Rough sets: Theoretical aspects of reasoning about data. Kluwer Dordrecht (1991)

[9] Salton, G.: Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley Longman Publishing Co., Inc. (1989)

[10] Skowron, A., Stepaniuk, J.: Tolerance approximation spaces. Fundamenta Informaticae **27** (1996) 245–253

[11] 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) 279–284

[12] Weiss, D.: A clustering interface for web search results in polish and english. Master's thesis, Poznan University of Technology, Poland (2001)

[13] Weiss, D.: Carrot2 Developers Guide, (http://www.cs.put.poznan.pl/dweiss/carrot/site/developers/manual/manual.pdf)

[14] Wroblewski, M.: A hierarchical www pages clustering algorithm based on the vector space model. Master's thesis, Poznan University of Technology, Poland (2003)

[15] Zamir, O., Etzioni, O.: Grouper: a dynamic clustering interface to web search results. Computer Networks (Amsterdam, Netherlands: 1999) **31** (1999) 1361–1374