Retriever: Improving Web Search Engine Results Using Clustering

Zhihua Jiang[1], Anupam Joshi[1], Raghu Krishnapuram[2], and Liyu Yi[2]
[1]Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD 21250
{joshi, zhjiang}@cs.umbc.edu

[2]Department of Mathematical and Computer Sciences
Colorado School of Mines
Golden, Colorado 80401
{rkrishna, lyi}@mines.edu

Abstract

Web search engines have become increasingly ineffective as the number of documents on the web have proliferated. Typical queries retrieve hundreds of documents, most of which have no relation with what the user was looking for. The objective of this work is to propose new techniques to cluster the results of a query from a search engine into groups. These groups and their associated keywords are presented to the user, who can then look into the URLs for the group(s) that s/he finds interesting. N-gram and vector space methods are used to create the dissimliarity matrix for clustering. We compare these distance metrics by clustering the data using a robust fuzzy algorithm and evaluating the results.

1 Introduction

Today, the WWW represents one of the largest, distributed, heterogeneous, semi-structured repositories of multimedia content. With the proliferation of Web servers and pages, users have experienced the exitement of the Web providing a large information repository, together with the frustration of trying to actually find anything useful in the morass of information. Typical search engine queries elicit hundreds, sometimes even thousands, of URLs from search engines, forcing the user to wade through them in order to find the URL(s) she needs. In large part, this can be attributed to the following:

- *The words involved in the search have multiple meanings.* For example, a user searching for cricket may be interested in the sport or the insect.
- *The user's desired search cannot easily be captured by keywords alone.* For example, a user looking for Web pages about work on mobile computing done by a particular group of people.

Notice that these problems are independent of how good the algorithms which associate keywords with the contents of a page are. Note further that these problems are common to any keyword type indexing scheme used for heterogeneous distributed digital library corpora.

One possible solution to this problem is to realize that the responses from search engines to a particular query can be broadly grouped into meaningful categories. If the user is shown these groups, possibly with some keyword type descriptions, they can select one (or more) which fit their perceived interests. Note that this is different from the site oriented grouping that some search engines present, typically in the form of a *similar pages from this site* link, since the aim here is to group together pages that originate from completely different servers. There has been prior work along these lines, such as that by Croft[3], and more recent work by Cutting *et al.*[4]. However, this work is in the context of general text collections.

The recent work of Etzioni *et al.* [8] proposes the notion of clustering Web search engine results. To the best of our knowledge, this is the only other work besides our own that seeks to cluster search engine results on the fly. They have proposed an algorithm called Suffix Tree Clustering (STC) to group together snippets from Web pages. Essentially, this algorithm uses techniques from literature which allow the construction of suffix trees in time linear in the number of snippets assuming that the number of words in each snippet can be bounded by a constant. Each node in this tree captures a phrase (some suffix of the snippet string), and has associated with it those snippets which contain it. These nodes are viewed as base clusters since they group documents having a phrase in common. Each cluster is assigned a score based on the number of URLs in the cluster as well as the size of the phrase that they have in common. In order to account for the fact that Web pages in the same group may have more than a phrase in common, they then create a graph which has as its vertices the clusters identified by the suffix tree. They define a binary similarity measure between the clusters which is set to 1 if at least half of the documents in each cluster are common to both. Vertices representing similar clusters are connected by an edge. They then run a connected component finding algorithm, and each connected component is identified as a grouping of documents that are similar.

The rationale behind clustering snippets rather than the Web documents themselves is essentially speed. Clearly, clustering the (much) shorter snippets takes much less time than clustering full pages, and makes it possible to create clusters on the fly in response to a user's search request.

Given that clusters are formed out of snippets the efficacy of the phrase commonality criterion used by STC is not clear. While commonality of phrases may be a valid criterion in grouping large document collections, it is not clear if it is quite as appropriate for grouping snippets. Snippets are typically the first few lines of (raw) HTML from the document. Once common words (e.g. HTTP related terms) (which are treated as stop words) are eliminated, what remains are essentially the heading of the page and the first sentence or two. Thus a phrase based approach will likely do no better than a word commonality based approach, and may even be detrimental. Further, the use of binary similarity definition between the initial clusters leads to arbitrary decisions on whether two clusters should be merged. For example, using 0.5 as the threshold would imply that clusters with 0.49 similarity would not be merged, whereas those with 0.51 similarity would. The aim of clustering the results would be better served by defining a *soft* similarity measure that takes continuous values in the 0 to 1 range. Fuzzy clustering thus seems to be appropriate in this context. Moreover, clustering snippets involves dealing with a significant amount of noise. One reason for the noise is that the responses from the search engines themselves are noisy - many of the URLs returned have little or no connection with the original query, nor are they a part of any coherent "group" of URLs. The other reason is the use of snippets – often the first few sentences of a document will fail to capture its essence. Thus the clustering technique used must be robust – able to handle significant noise and outliers.

In this paper, we describe a system to cluster search engine results based on a robust relational fuzzy clustering algorithm we have recently developed. We compare the use of Vector Space based and N-gram based dissimilarity measure to cluster the results from the Lycos search engine. We start by providing a brief background on stop word elimination and stemming, Vector space, and N-gram. We then describe our system, and discuss results from our experiments.

2 Background

2.1 Stop Word Elimination

"Stop words" are words with high frquency of occurence in common documents but without any special meaning in terms of identification and classification of the specific document. For example, the word "the" and "a" may be present frequently in most documents, but they are meaningless as keywords for identifying a document. We used a stop-word elimination algorithm to filter out insignificant words (such as HTML tags, articles) before generating the (dis)similarity matrix.

2.2 Vector Space

Vector space representations of document rely on first choosing k number of significant words from the group of documents as the base. A vector of length k is constructed as a reprsentative of individual document by setting the s_i to be 1 if the ith significant word appears in that document and 0 if not. We then could compare the (dis)similarity of two documents by using Jaccard measure shown below.

$$d(\mathbf{s}_1, \mathbf{s}_2) = \frac{\sum_i min(s_{1i}, s_{2i})}{\sum_i max(s_{1i}, s_{2i})} \tag{1}$$

We observe that in most languages, a word occurs in more than one format due to grammatical rules. For example, the word "fence" may appear as "fence" or "fencing" in different sentences. For document identicification and classfication, we should ignore such modifications of individual word and treat them as the same. Therefore, after stop word elimination, we also implement a stemming algorithm[2] to stem words in our snippets. In the previous example, we treat "fenced" or "fencing" as "fenc". In our experiments, 500 significant words from across the cleaned snippets are selected using inverted document frequency method[2]. Each snippet is then represented as 500 dimensional vector s, where s_i is the normalized frequency of the occurrence of the i^{th} significant word in the snippet. A dissimilarity matrix is then created by using Jaccard measure, which is later used in our clustering algorithm.

2.3 N-Gram

An n-gram is a character sequence of length n extracted from a document. It has been used in several automatic document indexing schemes [5, 6].

It is simple to generate n-grams out of a line of strings. For example, considering n=5 and the partial sentence "..relational fuzzy clustering..", the first several n-gram are "relat", "elati", "latio", and "ation". N-gram system is tolerant of minor spelling errors because of the redundancy introduced with the sliding n-gram approach, which identifies all unique n-grams in a document. An N-gram system can also achieve language independence by eliminating the language-dependent features such as stemming. We can calculate the (dis)similarity distance between any two snippets by looking for the ratio of the number of n-grams shared by the two over the minimal number of n-grams in each(Overlap coefficient). We also could generate the ratio by Dice coefficient, that is, 2*C/(A+B), where A and B are the number of n-grams in respective snippets and C is the number in common.

2.4 The Robust Fuzzy c Medoids Algorithm (RFCMdd)

We have recently proposed [7] an algorithm for fuzzy relational clustering based on the idea of identifying k-medoids. We call this algorithm Robust Fuzzy c-Medoids and abbreviate it as RFCMdd. The worst case complexity of RFCMdd is $O(n^2)$, but in practice it can be made linear and is an order of magnitude faster than the well known RFCM algorithm[1]. Since we use a fuzzy algorithm, we are able to handle partial membership situations common in this task – in other words when the same URL may belong to two different groups but to different "degrees". Moreover, our algorithm is highly robust and thus able to handle noise much better than traditional clustering approaches. Note that the data we cluster here (snippets) are highly noisy to begin with in terms of representing the actual documents. In addition, noise is also introduced in our distance generation measures.

3 System Design

Our system, called Retriever, is designed as a *client–proxy–server* system, and its architecture is illustrated in Figure 1. The proxy server is a Perl-based system that connects to the Lycos search engine using an internal API that Lycos allows us to use. The search term(s) entered by the user are passed onto the proxy via a CGI program. After transforming the query to a request in the Lycos API format, the proxy server then passes the search term(s) to the Lycos search engine. The results that arrive back from Lycos are first trapped by the proxy and are saved to a file. The clustering routine then works on them, and presents the grouped results to the users.

In response to a query, the system fist returns a page (Figure 3) quite similar to the page returned from any common search engine. It contains a brief list of titles, urls, and their descriptions. If users can easily locate the links they want from among the first few paragraphs, they may simply click the link to the destination. Otherwise, they may click the button on the upper right corner, labeled "Clusters", to see the grouped results. After the button is clicked, another web browser window will pop up to show the results in frames (Figure 4). Users may browse each cluster to pick out topics that they are interested in by following the link in the left frame. This causes the corresponding group of URLs to be displayed in the right frame (Figure 5).

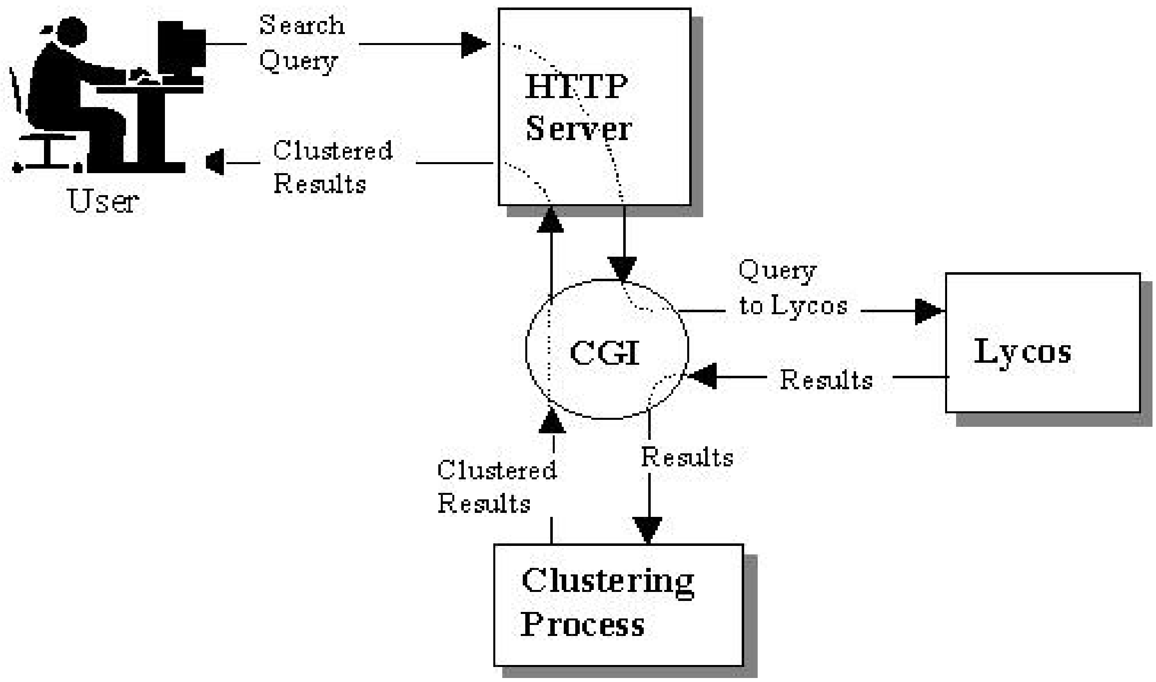

Figure 1: System Architecture

Two different methods for the calculation of distance matrix are included in our demonstration site. The first one associated with "VectorSpace Search" button performs vector space based RFCMdd algorithm. It utilizes TF/IDF method to generate the distance matrix. The second one attached to "N-Gram Search" button implements n-gram based RFCMdd algorithm.

4 Experimental Results

We compare the efficiency of n-gram based method for distance matrix generation (using n=5) vs vector space based method. Our RFCMdd algorithm is used for clustering the snippets. Several experiments have been run for the comparison. The keywords used in the queries are *salsa*, *moon river*, and *mobile robots* respectively. These are chosen so that the returned documents will fall into groups. For example, documents returned in response to the query *salsa* should be groupable into those referring to the *salsa*, and those referring to the music/dance, etc. For each query, we record the number of URLs returned by Lycos search engine, the number of clusters formed by the algorithm, and the number of URLs in each cluster (as a percentage of the total).

One of the authors(ZJ), prior to the start of the experiments, was asked to "hand cluster" the URLs into groups. We start by noting that clustering algorithm results in a larger number of groups than the human. In part this is due to "meta knowledge" that humans have about the structure of the sites which neither our approach nor other works can take into account. For example, on searching on the keywords *washington post*, the human groups all URLs from the washington post newspaper into one category. However, the clustering algorithm partitions these URLs into several

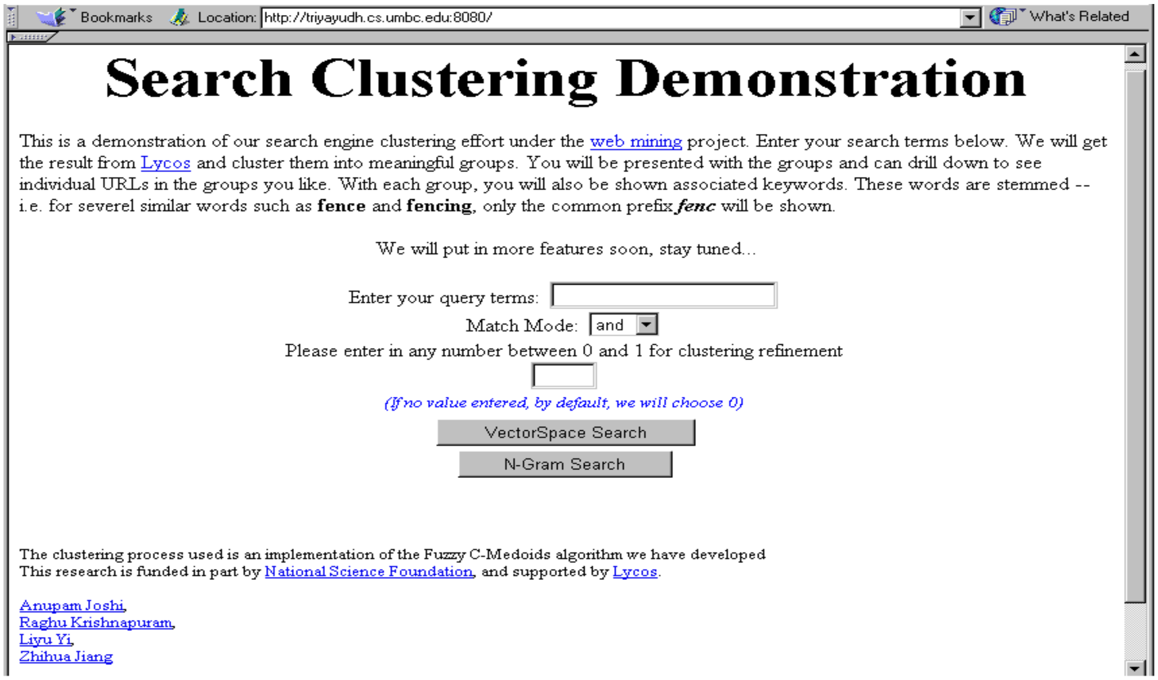

Figure 2: System Interface

Figure 3: First Index Page

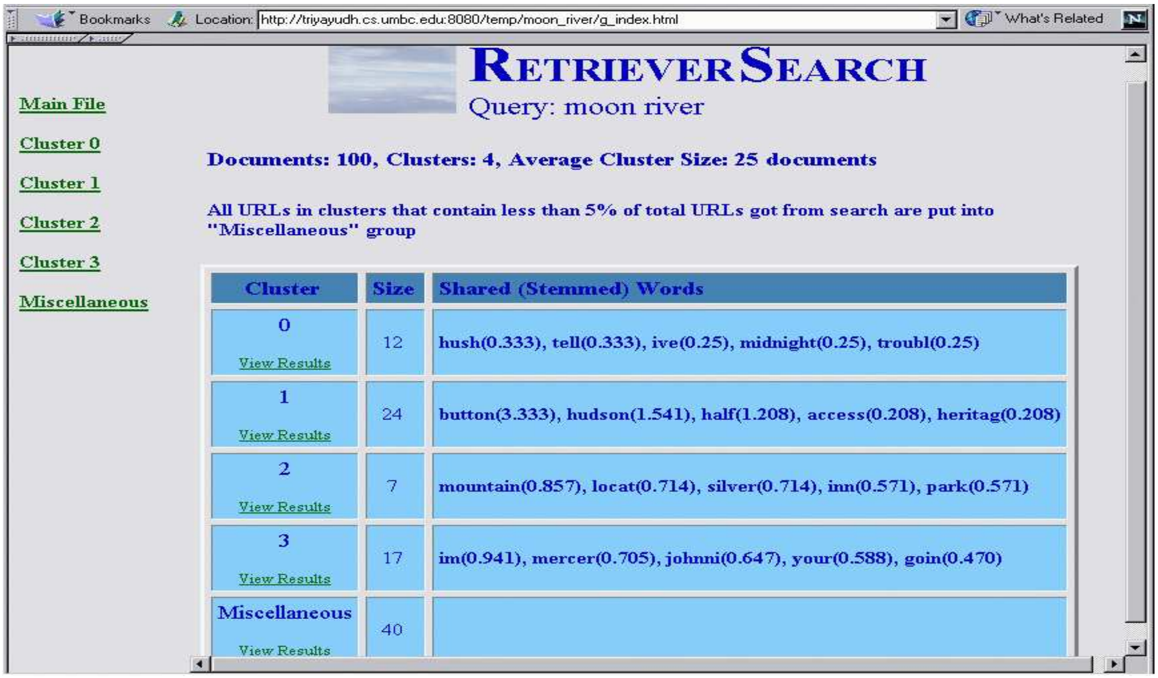

Figure 4: Group Index Page

Figure 5: Cluster Example Page

clusters corresponding to sports pages, lifestyle pages, politics etc. This is to be expected since the "keywords" occurring in the snippets for these sites are different.

The results for the experiments dealing with the three queries mentioned earlier are summarized in the following tables. Table 1 shows the number of URLs returned by the search engines in response to the particular query, and the number of clusters produced by different measures. Note that since the (dis)similarity measure underlying is different for these two experiment, a difference in the number of clusters is not unexpected. Table 2 shows how the URLs are distributed across the clusters. Note that while we show the cluster numbers across different methods (n-gram vs vector space) in the same row in the table – the cluster numbers are simply arbitrary labels assigned by the clustering algorithm. Thus Cluster 1 of vector space based test has nothing to do with cluster 1 of n-gram based one, and so on. The significance of this table is in showing the distribution of URLs within the clusters generated by the same algorithm.

In the results for *mobile robots* search, we observe that Cluster 0, 15, 18, 19, and 20 created by vector space based RFCMdd algorithm have the largest concentration of URLs, followed by clusters 5, 7, 8, 11, and 16. The remaining 11 clusters contain few URLs. The n-gram based RFCMdd algorithm leads to Clusters 0, 1, and 2 having a majority of the URLs.

For the query on *salsa*, vector space based RFCMdd creates 4 dominant clusters – 0, 4, 16, and 20, whereas n-gram RFCMdd creates 2 dominant clusters, 0 and 1. For the *moon river* search, vector space based RFCMdd creates one major cluster, and n-gram RFCMdd two.

Tables 3– 8 illustrate the keywords associated with the clusters. In the *mobile robots* query for example, some of the clusters deal with a course in the CS department at the Brandies University, and others deal with mobile agents, AI, and control. For Salsa, the clusters pertain to the music, the dance, the sauces, and the OS companion (to NACHOS, the instructional OS system). Finally for moon river, the clusters deal with various resorts, besides of course the famous waltz.

From the comparison between vector space based RFCMdd and n-gram based RFCMdd, we observe that normally, n-gram based RFCMdd generates fewer clusters than vector space based one. In addition, the urls are distributed more narrowly across clusters. In other words, n-gram based RFCMdd algorithm seems to provide better results.

We also evaluate the effect of the value of n in n-gram based RFCMdd algorithm in order to discover the best value of n for our RFCMdd algorithm. Again, *moon river*, *mobile robots*, and *salsa* are used as examples. Table 9 shows the diferent number of clusters generated given different value of n and Table 10 illustrates the number of urls in each cluster and url distributions. We notice that in the query of mobile robots, when $n = 3$, n-gram RFCMdd create 9, the most number of clusters, which is nearly half the number of clusters generated by vector space based RFCMdd. However, when $n = 2$, the least number of clusters are generated. The reason is that when the length of gram is too small, say n=1 or n=2, it is difficult to tell apart two different snippets due the fact that the possibility of the same gram appearing in different snippets becomes larger.

We also did the similar comparison on query salsa and moon river. The results do not show dramtic change of the number of clusters given the different length of grams, as in the experiment of mobile robots. But we observe that when n=5 and n=6, the number of clusters will stay similar and the major distribution of urls in clusters will narrow down to 1-3 clusters. This observation is in consonance with prior work in IR, where 5 grams have been shown to be useful in document identification tasks.

Finally, we use an implementation of Etzioni et al.'s system (http://zhadum.cs.washington.edu/)

based on Suffix Tree Clustering (STC). We present results of the comparison in Table 11. The keyword list generated from STC is in Table 12– 14 and url distribution is in Table 15.

For Zamir and Etzioni's STC algorithm, we present the keywords/phrase with the associated strength as reported by their algorithm in the tables. For our n-gram based RFCMdd algorithm, we present the stemmed keywords most often associated with the cluster, as well as its normalized frequency of occurance. For purposes of displaying these tables within page confines, we have sometimes presented only a part of a phrase or a long word, and indicated that by placing a *.

Based on these preliminary experiments, it seems that n-gram based (dis)similarity measure is more suitable to this application than vector space based measure. It leads to a fewer number of more focused clusters and seems to be closer to the "hand clustering" results of ZJ. We note that the vector space approach leads to results similar to the STC methods, and so the n-gram based clustering seems to do better than STC as well.

5 Conclusions

In this paper, we have presented a system which seeks to improve the process for finding relevant URLs for the users. In particular, the results returned from a search engine (in our case, Lycos), are clustered on the fly into groups, and these groups and their associated keywords are presented to the users. The user can then chose to examine URLs in one or more of these groups based on the keywords. We have used a new robust relational fuzzy clustering algorithm based on the idea of medoids which we have recently developed (RFCMdd). The worst-case complexity of the algorithm is $O(n^2)$, which happens while updating the medoids in each iteration. In addition, we introduce n-gram method and vector space method to generate the (dis)similarity distance matrix to enhance the performance of our algorithm. Our preliminary results show that the algorithm gives good results on Web snippets. The n-gram based approach seems to perform better than the vector space based approach, and as well as Etzioni et al.'s STC algorithm [8]. Note that we draw snippets from Lycos, whereas STC draws on Metacrawler. While this can explain the minor variation in clustering seen between STC and vector space approach, it does not detract from the apparently better performance of the n-gram approach. Moreoever, our approach captures the overlapping clusters idea (a URL can belong to more than one group to different degrees) more elegantly and does not force the user to make an arbitrary "binary" choice of declaring two groups to be similar. Moreover, our algorithm is robust, i.e. not sensitive to noise and outliers which are a common occurrence in this domain. We realize of course that in order to achieve speed (clustering the results from the search engine as they come back), we are sacrificing accuracy by clustering only the snippets rather than the documents themselves. In ongoing research, we are looking to explore new similarity measures between snippets that would better capture their closeness than word or phrase commonality. We are also looking at further robustifying the clustering process and strengthening its outlier detection. This will allow us to either explicitly flag outliers to the user or suppress their display. Finally, we would like to expand our experiments to allow humans to subjectively compare the clustering results across various approaches.

moon river	Vector Space	Ngram
number of URLs	100	100
number of clusters	21	2
mobile robots	**Vector Space**	**Ngram**
number of URLs	200	200
number of clusters	21	5
salsa	**Vector Space**	**Ngram**
number of URLs	200	200
number of clusters	21	3

Table 1: Number of URLs clustered by the two methods for three queries

Acknowledgements

Partial support of this work by grants from National Science Foundation (IIS 9800899 to Krishnapuram, IIS 9801711 to Joshi) is gratefully acknowledged.

References

[1] J. C. Bezdek, R. J. Hathaway, and M. P. Windham. Numerical comparison of the rfcm and ap algorithms for clustering relational data. 24:783–791, 1991.

[2] J. Chen, A. Mikulcic, and D. H. Kraft. An integrated approach to information retrieval with fuzzy clustering and fuzzy inferencing. In O. Pons, M. Ampara Vila, and J. Kacprzyk, editors, *Knowledge Management in Fuzzy Databases*. Physica Verlag, Heidelberg, Germany, 1998.

[3] W.B. Croft. *Organizing and Searching Large Files of Documents*. PhD thesis, Cambridge University, 1978.

[4] D. Cutting, D. Krager, J. Pedersen, and J. Tukey. Scatter/gather: A cluster based approach to browsing large document collections. In *Proc. 16^{th} ACM SIGIR Conference*, pages 318–329, 1992.

[5] Marc Damashek. U.s. patent number 5,418,951. 1995.

[6] R. D'Amore and C. Mah. One-time complete indexing of text: theory and practice. In *Proceedings 8th Internation ACM Conference on Research and Development in Information Retrieval. ACM Press*, 1985.

[7] R. Krishnapuram, A. Joshi, and L. Yi. A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, 1999. (to appear).

[8] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Proc. ACM SIGIR '98*, 1998.

moon river	Vector Space		Ngram	
/	**Absolute**	**percentage**	**Absolute**	**percentage**
C0	1	0.010	39	0.390
C1	2	0.020	61	0.610
C2	2	0.020		
C3	2	0.020		
C4	2	0.020		
C5	1	0.010		
C6	1	0.010		
C7	1	0.010		
C8	2	0.020		
C9	1	0.010		
C10	3	0.030		
C11	56	0.560		
C12	1	0.010		
C13	2	0.020		
C14	8	0.080		
C15	3	0.030		
C16	1	0.010		
C17	3	0.030		
C18	1	0.010		
C19	5	0.050		
C20	2	0.020		

mobile robots	Vector Space		Ngram	
/	**Absolute**	**percentage**	**Absolute**	**percentage**
C0	70	0.350	28	0.140
C1	1	0.005	120	0.600
C2	4	0.020	45	0.225
C3	3	0.015	1	0.005
C4	1	0.005	6	0.030
C5	6	0.030		
C6	1	0.005		
C7	5	0.025		
C8	8	0.040		
C9	1	0.005		
C10	1	0.005		
C11	8	0.040		
C12	1	0.005		
C13	1	0.005		
C14	2	0.010		
C15	29	0.145		
C16	5	0.025		
C17	1	0.005		
C18	15	0.075		
C19	22	0.110		
C20	15	0.075		

salsa	Vector Space		Ngram	
/	**Absolute**	**percentage**	**Absolute**	**percentage**
C0	81	0.405	25	0.125
C1	1	0.005	163	0.815
C2	4	0.020	12	0.060
C3	3	0.015		
C4	19	0.095		
C5	4	0.020		
C6	1	0.005		
C7	12	0.060		
C8	13	0.065		
C9	1	0.005		
C10	1	0.005		
C11	9	0.045		
C12	4	0.020		
C13	1	0.005		
C14	1	0.005		
C15	2	0.010		
C16	14	0.070		
C17	1	0.005		
C18	1	0.005		
C19	8	0.040		
C20	19	0.095		

Table 2: number of urls in each cluster and url distribution

Vector Space			Key words		
C0	manag 2	colleg 1	bus444 1	bus670 1	graduat 1
C1	dream 2.5	affair 1	river 1	experienc 0.5	extraordinari 0.5
C2	fogerti 2	john 2	blue 1	releas 1	album 0.5
C3	cassett 2.5	album 1	kc 1	lp 1	passport 1
C4	polit 1.5	experi 1	concern 1	butler 1	adventur 1
C5	histor 2	epicent 1	crash 1	border 1	lt 1
C6	re 4	cocteaux 2	onion-paper 2	quai 2	thingi 2
C7	f1dylsj1 1	bmhipj 1	biqbf 1	william 0	welcom 0
C8	quot 1.5	favorit 1	human 1	caus 1	climat 0.5
C9	stock.landscape.html 5	brush 1	cabin 1	greenbrier 1	gsmnp 1
C10	mt 1.666	castl 1	demesn 1	photo 1	pleasant 1
C11	river 1.214	m2n 0.714	music 0.428	network 0.196	jamaica 0.178
C12	million 3	block 1	feet 1	cartouch 1	egyptian 1
C13	bahama 4.5	bahia 1.5	california 1.5	baja 1.5	charlwood 1.5
C14	page 1	link 0.875	alamo 0.875	fish 0.5	hobbi 0.5
C15	banner 1	game 0.666	god 0.666	heart 0.666	love 0.666
C16	adventur 0	zone 0	world 0	william 0	welcom 0
C17	im 1.333	die 1	au 0.666	live 0.666	schmidt 0.666
C18	brigl 4	artist 2	gayl 2	exclus 1	collect 1
C19	link 1	artwork 0.6	local 0.6	cd 0.6	advertis 0.4
C20	bai 1.5	georgian 1	club 0.5	face 0.5	feng 0.5

Table 3: Clustering of Moon River using Vector Space

Vector Space			Key words		
C0	lab 1	page 0.828	system 0.671	ee 0.628	homework 0.542
C1	abet 0	zurich 0	yunece.nps.navy.mil 0	ymposium 0	www.cs.brandeis.edu 0
C2	avail 1.5	ftp 0.75	anonym 0.75	via 0.75	hobbes.jsc.nasa.gov 0.5
C3	cours 1.333	data 1	electr 1	fusion 0.666	master 0.666
C4	abet 0	zurich 0	yunece.nps.navy.mil 0	ymposium 0	www.cs.brandeis.edu 0
C5	imag 1	process 0.666	motion 0.5	descript 0.333	homebuilt 0.333
C6	fight 1	freedom 1	god 1	hand 1	countri 1)
C7	artifici 1.4	usa 0.8	intellig 0.6	appli 0.4	center 0.4
C8	design 0.875	program 0.625	confer 0.5	proceed 0.5	real-time 0.375
C9	flamingo 1	edu 1	yunece.nps.navy.mil 0	ymposium 0	www.cs.brandeis.edu 0
C10	abet 0	zurich 0	yunece.nps.navy.mil 0	ymposium 0	www.cs.brandeis.edu 0
C11	agent 0.75	algorithm 0.625	autonom 0.625	circl 0.25	graph 0.25
C12	abet 0	zurich 0	yunece.nps.navy.mil 0	ymposium 0	www.cs.brandeis.edu 0
C13	abet 0	zurich 0	yunece.nps.navy.mil 0	ymposium 0	www.cs.brandeis.edu 0
C14	januari 1.5	colloquium 1	graduat 1	survei 1	exampl 0.5
C15	control 1.310	system 1.241	engin 0.689	ee 0.586	intellig 0.448
C16	network 0.8	sparc 0.8	autonom 0.6	server 0.6	dec 0.4
C17	abet 0)	zurich 0	yunece.nps.navy.mil 0	ymposium 0	www.cs.brandeis.edu 0
C18	upgrad 1.333	project 1.066	den 0.666	mrv4 0.533	softwar 0.4
C19	comput 0.818	scienc 0.681	faq 0.409	institut 0.363	intellig 0.363
C20	navig 0.533	plan 0.533	vision 0.533	perform 0.266	control 0.266

Table 4: Clustering of Mobile Robots using Vector Space

Vector Space			Key words		
C0	hot 0.617	sauc 0.382	chile 0.333	food 0.222	gourmet 0.197
C1	ab 0	zarama 0	youv 0	youll 0	wish 0
C2	love 1	edi 0.75	scene 0.5	clubsnbsp 0.25	formerli 0.25
C3	desktop 1.333	import 1	window 1	guid 0.666	complex 0.666
C4	danc 1.947	lesson 0.473	merengu 0.368	cha 0.315	dj 0.315
C5	applic 2	data 1	intranet 1	compil 1	code 0.75
C6	kit 3	starter 3	human 2	resourc 2	cont 1
C7	special 1.083	post 0.583	juli 0.5	violet 0.416	mix 0.333
C8	recip 1.076	tomato 0.692	cup 0.538	garlic 0.307	dice 0.307
C9	constructio 1	zarama 0	youv 0	youll 0	wish 0
C10	nikki 2	talk 2	youv 0	youll 0	wish 0
C11	post 0.666	e 0.555	festiv 0.555	latino 0.444	dj 0.444
C12	oz 1.25	pure 1.25	bright 1	il 1	ballroom 0.75
C13	colleg 1	environment 1	error 1	youll 0	wish 0
C14	ab 0	zarama 0	youv 0	youll 0	wish 0
C15	tango 2.5	miguel 2	alkeet 2	jatkonbsp 1	kaari 1
C16	music 1.357	latin 0.571	dj 0.357	todai 0.285	danc 0.214
C17	ab 0	zarama 0	youv 0	youll 0	wish 0
C18	no 2	todo 2	dej 1	compartir 1	wish 0
C19	tast 1	locat 0.5	insan 0.375	mexico 0.375	click 0.375
C20	post 1.421	midi 0.421	septemb 0.368	file 0.368	januari 0.263

Table 5: Clustering of Salsa using Vector Space

Ngram			Key words		
C0	m2n 0.615	music 0.384	link 0.256	bahama 0.230	site 0.205
C1	river 1.180	m2n 0.262	music 0.213	jamaica 0.163	song 0.147

Table 6: Clustering of Moon River using Ngram

Ngram			Key words		
C0	ee 0.5	system 0.428	design 0.25	cours 0.25	faq 0.214
C1	system 0.55	lab 0.516	page 0.4	ee 0.35	control 0.325
C2	system 0.466	upgrad 0.288	faq 0.2	page 0.2	control 0.2
C3	materi 2	benefit 2	archiv 2	hold 1	www.cs.brandeis.edu 0
C4	artifici 0.666	autonom 0.666	usa 0.666	motion 0.5	agent 0.333

Table 7: Clustering of Mobile Robots using Ngram

Ngram			Key words		
C0	latin 0.2	music 0.2	oz 0.2	pure 0.2	bright 0.16
C1	post 0.294	hot 0.257	danc 0.233	sauc 0.190	chile 0.165
C2	hot 0.75	recip 0.5	vermont 0.333	asado 0.25	product 0.25

Table 8: Clustering of Salsa using Ngram

moon river	2	3	4	5	6
number of URLs	100	100	100	100	100
number of clusters	2	3	3	2	2
mobile robots	**2**	**3**	**4**	**5**	**6**
number of URLs	200	200	200	200	200
number of clusters	2	9	8	5	4
salsa	**2**	**3**	**4**	**5**	**6**
number of URLs	200	200	200	200	200
number of clusters	4	4	3	3	4

Table 9: Number of clusters generated in each experiment as the n-gram length is varied

moon	2		3		4		5		6	
river	**Absolute**	**percentage**	**Absolute**	**percentage**	**Absolute**	**percentage**	**Absolute**	**percentage**	**Absolute**	**percentage**
C0	25	0.250	1	0.010	13	0.130	39	0.390	47	0.470
C1	75	0.750	43	0.430	86	0.860	61	0.610	53	0.530
C2			56	0.560	1	0.010				
mobile	**2**		**3**		**4**		**5**		**6**	
robots	**Absolute**	**percentage**	**Absolute**	**percentage**	**Absolute**	**percentage**	**Absolute**	**percentage**	**Absolute**	**percentage**
C0	105	0.525	9	0.045	7	0.035	28	0.140	42	0.210
C1	95	0.475	3	0.015	16	0.080	120	0.600	18	0.090
C2			48	0.240	47	0.235	45	0.225	131	0.655
C3			6	0.030	17	0.085	1	0.005	9	0.045
C4			4	0.020	15	0.075	6	0.030		
C5			3	0.015	68	0.340				
C6			122	0.610	27	0.135				
C7			4	0.020	3	0.015				
C8			1	0.005						
salsa	**2**		**3**		**4**		**5**		**6**	
/	**Absolute**	**percentage**	**Absolute**	**percentage**	**Absolute**	**percentage**	**Absolute**	**percentage**	**Absolute**	**percentage**
C0	73	0.365	2	0.010	7	0.035	25	0.125	40	0.200
C1	25	0.125	182	0.910	183	0.915	163	0.815	152	0.760
C2	28	0.140	15	0.075	10	0.050	12	0.060	1	0.005
C3	74	0.370	1	0.005					7	0.035

Table 10: Url distribution and number of urls in each clusters as the n-gram length is varied

moon river	Ngram RFCMdd	STC
number of URLs	100	186
number of clusters	2	16
mobile robots	**Ngram RFCMdd**	**STC**
number of URLs	200	208
number of clusters	5	16
salsa	**Ngram RFCMdd**	**STC**
number of URLs	200	180
number of clusters	3	16

Table 11: Number of URLs clustered by the two methods for three queries

STC		Key words/phrases							
C0	zulu* (0.22)	History (0.52)	City (0.43)	Pictures (0.35)					
C1	Georgian * (0.67)	designed* (0.67)	adjacent (0.67)	Moon River (0.67)					
C2	Half Moon (0.41)	crossing* (0.18)	Moon River* (0.18)	Hudson (0.27)					
C3	classroommc* (0.67)	emc (1)	Sons (0.50)	Stand (0.50)					
C4	Complete Listing (0.29)	RegionUS (0.29)	Located (0.35)	Travel (0.24)					
C5	William H (0.23)	book (0.32)	Publication (0.23)	Chapter (0.23)					
C6	MOON RIVER* (0.75)	true (0.63)	pure (0.50)	stallion (0.50)	Polish (0.37)	Crabbet (0.37)	lines (0.37)	proven (0.37)	herd (0.37)
C7	natural (0.36)	American (0.36)	run (0.29)	Mountain (0.29)					
C8	Sailor Moon (0.63)	series (0.63)	Story (0.50)	people (0.37)					
C9	HISTORIC (0.50)	Property (0.40)	newspapers (0.40)	Press (0.40)	Half (0.40)				
C10	SONG (0.67)	music (0.67)	Lyrics (0.41)						
C11	County (0.78)	Valley (0.67)							
C12	Earth and (0.57)	Earth (1)							
C13	Lyrics (1)								
C14	day (1)								
C15	Miscellaneous								

Table 12: Clustering of Moon River Responses by STC

STC			Key words		
C0	Catering Menu* (0.63)	Housing* (0.50)	Reference Tools (0.50)	Administrative Guide (0.50)	
C1	Salsa (0.75)	Arts (1)	clubs (0.50)		
C2	Hot Sauce (0.31)	Food (0.25)	Condiments (0.25)	Sauce ¿ Salsa (0.25)	salsas (0.37)
C3	Latin Dance (0.33)	Salsa Music (0.25)	Dance Lessons (0.17)	reviews (0.29)	
C4	guide to (0.17)	clubs and (0.17)	latin music (0.14)	music in (0.14)	dance (0.51)
C5	Fiery foods (0.29)	Gourmet Salsa (0.29)	chips (0.35)	love (0.29)	
C6	Mexican food (0.20)	Mexican (0.50)	Mexico (0.35)	chile (0.30)	
C7	Music (0.77)	Latin (0.55)	dance (0.48)		
C8	dance (1)	Latin (0.37)			
C9	HOT SALSA (0.40)	Hot (1)	Food (0.35)		
C10	food (1)				
C11	Chiles (50)	Los (50)	Mild (40)	Hot (40)	Sauces (40)
C12	guide (100)	Foods (38)			
C13	Recipes (100)				
C14	salsas (100)	Hot (45)	Food (36)	Sauces (36)	Sauce (36)
C15	Miscellaneous				

Table 13: Clustering of Salsa Responses using STC

STC			Key words					
C0	Robotics (0.42)	Engineering (0.96)	robots (0.96)	research (0.42)				
C1	Current Projects* (0.20)	Robert Albrecht (0.16)	Current (0.44)	Abstract (0.40)	robots (0.72)	research (0.40)		
C2	FAQ (0.44)	Resources in VR (0.25)	knowledge base (0.25)	Newsgroups (0.50)	Robotics (0.37)			
C3	autonomous mobile* (0.48)	autonomous (1)	robotics (0.41)					
C4	Mobile Robotics* (0.89)	Robot (0.77)	Research (0.44)					
C5	Robotics (0.67)	Institutes (0.67)	Robot (1)	research (0.66)	Intelligent (0.50)	Laboratory (0.50)		
C6	Robot Laboratory (1)	Mobile Robot Lab (0.75)	Robotics (0.62)	research (0.50)	Engineering (0.37)			
C7	Simulation of (0.24)	Autonomous robots (0.20)	autonomous sys* (0.20)	Principles of (0.16)	Robotics (0.36)			
C8	intelligent mobile* (1)	Robotics (0.37)						
C9	Robotics (1)	robots (0.72)						
C10	AI* (0.42)	intelligence (0.83)	artificial (0.67)	robots (75)	Robotics (0.41)	research (0.41)	autonomous (0.41)	Laboratory (0.41)
C11	EE* (0.86)	robots (0.85)	Robotics (0.71)	research (0.42)	departments (0.42)	Lab (0.42)		
C12	Intelligence Lab* (0.25)	machine (0.38)	Laboratories (0.31)	performance (0.31)	Robotics (0.75)	robots (0.75)		
C13	research (1)	robots (0.69)	Robotics (0.58)					
C14	Mobile Robotics (1)	robots (0.77)	Project (0.37)					
C15	Miscellaneous							

Table 14: Clustring of Mobile Robots Responses using STC

moon river	Ngram RFCMdd		STC	
/	**Absolute**	**pecentage**	**Absolute**	**percentage**
C0	39	0.390	22	0.11
C1	61	0.610	18	0.09
C2			41	0.21
C3			6	0.03
C4			16	0.08
C5			15	0.08
C6			10	0.05
C7			9	0.05
C8			9	0.05
C9			11	0.06
C10			8	0.04
C11			7	0.04
C12			7	0.04
C13			8	0.04
C14			8	0.04
C15			63	0.33

mobile robots	Ngram RFCMdd		STC	
/	**Absolute**	**percentage**	**Absolute**	**percentage**
C0	28	0.140	15	0.07
C1	120	0.600	112	0.54
C2	45	0.225	20	0.10
C3	1	0.005	12	0.06
C4	6	0.030	17	0.08
C5			6	0.03
C6			20	0.10
C7			6	0.03
C8			26	0.12
C9			21	0.10
C10			11	0.05
C11			17	0.08
C12			10	0.05
C13			12	0.06
C14			7	0.03
C15			45	0.22

salsa	Ngram RFCMdd		STC	
/	**Absolute**	**percentage**	**Absolute**	**percentage**
C0	25	0.125	48	0.27
C1	163	0.815	9	0.05
C2	12	0.060	10	0.05
C3			21	0.12
C4			7	0.04
C5			6	0.03
C6			19	0.10
C7			26	0.14
C8			18	0.10
C9			21	0.12
C10			12	0.07
C11			18	0.10
C12			13	0.07
C13			13	0.07
C14			13	0.07
C15			53	0.29

Table 15: Number and Percentage of URLs in each cluster