KhojYantra: An Integrated MetaSearch Engine with Classification, Clustering and Ranking

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

Rajeev Kumar Mishra

to the

Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

February, 2000

Certificate

This is to certify that the work contained in the thesis entitled "*KhojYantra: An Integrated MetaSearch Engine with Classification, Clustering and Ranking*", by *Rajeev Kumar Mishra*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

February, 2000

(Prof. T.V.Prabhakar)
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.

Abstract

The present search engines generally return a long ordered list of results which the users are forced to sift for getting relevant documents. The Information Retrieval Community has explored various methods for information presentation, which help users to separate interesting information from the set of retrieved documents. The *ranked list* is a well known and widely accepted technique for information presentation. *Clustering and Classification* are also explored as successful methods for organizing retrieved results, but these techniques are yet to be deployed as part of major search engines.

It is envisaged that the search results can be greatly improved by integrating techniques, like ranking,classification and clustering, in the mechanism of the information presentation.

This thesis explores how these three techniques can be integrated into search engine technology to achieve this goal. We introduce *KhojYantra*, an Integrated MetaSearch Engine with Classification, Clustering and Ranking. Some of the prominent features of the *KhojYantra* includes better classification scheme to divide the retrieved documents into relevant and general group, its way of clustering the documents and its technique of prioritizing the documents and the clusters according to their relevancy. *KhojYantra* also provides an easy to browse interface which along with the above techniques enhances the retrieval effectiveness of traditional ranked list. We design the system as a secondary search engine to reduce the consumption of resources by using the index of primary search engines as well as getting an advantage of obtaining higher coverage of indexed web documents.

Acknowledgements

My greatest thanks are due to my thesis supervisor, Dr. T.V.Prabhakar for his invaluable advice, enthusiastic inspiration, and never ending guidance. He was always there to help me whenever I needed it. He not only showed me how to conduct the thesis work, but also taught me how to be a human being with great integrity. My regards to Hannes Marais, Compaq Systems Research Center (SRC), for his invaluable help in the WebL language.

I would like to thank the faculty members of Computer Science Department for all the knowledge they have given to me. I would also like to thank the technical staff of the department for providing such a congenial working enviroment.

I am very fortunate to have such wonderful friends, who made my stay at IIT Kanpur a memorable experiance. I thank all of them, specially Godha, Vikas, Venkata, Major Ulhas and Rahul for their precious suggestions and everlasting help. I am greatly indebted to my seniors Atul and Kshitiz for helping me out in crucial situations. I am forever obligated to my parents and other members in my family for their perdurable encouragement to me, and unwavering faith in my ability to succeed. Finally I would like to give my special thanks to the great Almighty, Without His blessings, I wouldn't be able to stand where I am.

Contents

Acknowledgements **i**

1 Introduction **1**

1.1 Motivation 2

1.2 The Scope of Our Work 2

1.3 Organization of the Report 3

2 Background and Related Work **4**

2.1 Background 4

2.1.1 Search Tools 4

2.1.2 Ranking 7

2.1.3 Classification 7

2.1.4 Clustering 7

2.2 Document Representation 12

2.2.1 Defining a Vector Space 12

2.2.2 Controlling Dimensionality 17

2.2.3 Related Work 18

3 KhojYantra - Design **20**

3.1 Design Goals 20

3.2 Architecture of KhojYantra 22

3.3 User Interface 22

3.4 Obtaining initial set of Documents 23

3.5 Documents Processing 24

3.5.1 Duplicate Removal 24
3.6 Classification of Initial Set of Documents 25
3.6.1 Intuitive Justification of this Approach 25
3.6.2 Processing of documents in the *Relevant Groups* 26
3.6.3 Processing of Documents in the General Group 26
3.6.4 Results Presentation 28

4 Implementation **30**
4.1 User Interface 30
4.2 Multi-Engine Search Architecture 30
4.3 Document Processing and Classification of Initial Set 31
4.4 Processing of Relevant Document Set 33
4.4.1 SiteBased Clustering 33
4.4.2 Ranking Algorithm 34
4.4.3 Result Presentation 35
4.5 Processing of General Document Set 35
4.5.1 SiteBased Clustering 36
4.5.2 Ranking Algorithm 36
4.5.3 Obtaining Data Matrix 37
4.5.4 Similarity Measures 38
4.5.5 Clustering Algorithm 39
4.5.6 Result Presentation 41

5 Conclusion and Future Work **43**
5.1 Experiments 43
5.2 Directions for Further Work 44

Bibliography **47**

List of Tables

2.1 Sample Document No.1 14
2.2 Sample Document No.2 14
2.3 A simple vector representation of the sample documents 14
2.4 Sample Document No.1 15
2.5 Sample Document No.2 15
2.6 A vector representation of the stemmed versionof the sample documents 15
2.7 Sample List of StopWords 17

List of Figures

2.1 Example of a dendrogram 9
2.2 Single linkage and Complete linkage distances 11
2.3 Group average distance 11
2.4 Initial Document Processing and Representation 13

3.1 The Architecture of KhojYantra 23

4.1 KhojYantra's Interface 31
4.2 Multi-Engine Search Architecture 32
4.3 Example showing the result for keyword: *Introduction of Search Engines* 42

5.1 Search Results for the keyword "XML databases" 45
5.2 Search Results for the keyword "Complexity Theory" 46

Chapter 1

Introduction

In the 20th century, it is not wrong to say that mankind has faced the most rapid development of technology ever in the entire history of human beings. Apart from all other inventions that have contributed to the improvement of our technology and way of life, the impact created by World Wide Web to the people is unbeatable.

World Wide Web emerges as a wide source of online information and created a revolution in the area of information technology. One can access volumes upon volumes of resources and information on practically any topic. The sheer quantity of information available can make searching for specific information quite a daunting task. This forced us to develop search tools which help users to get the information according to their norms. These tools are named as search engines. There are quite a number of Search Engines available today. Each of them have their own databases containing all the documents in a particular form.The documents are usually collected by a program called a "web robot", which goes to the web and collects pages by following links. Maintenance of the database is essential since the WWW is growing rapidly every year.

Even though, the Search Engine is very useful and helpful in getting information, this technology is still young. There are yet a number of problems with them, which again becomes worse as the WWW is continuously growing in a very rapid manner.

1.1 Motivation

With the increasing availability of on-line information, users now have access to thousands of information sources and millions of documents. Simultaneously, there has been a strong focus on Search Engines as a means of extracting relevant information from this on-line huge information database. Given that much of on-line information, the question that comes to our mind is how to facilitate the access of information as per the specific requirements of a user ?.

Conventional document retrieval systems return long lists of ranked documents that users are forced to sift through to find relevant documents. The majority of today's Web search engines (e.g., Excite, Altavista) follow this paradigm. The notoriously low precision of Web search engines coupled with the ranked list presentation makes it hard for users to find the information they are looking for. Unfortunately, these methods for accessing information are quickly becoming more inadequate as the amount of on-line information continues to grow at an unprecedented rate.

Thus a system is required which is able to help user to locate the relevant information more quickly and therefore more effectively. As the size of on-line information is growing in a unpredictable way, it is also impossible for a single search engine to index all the pages present in the WWW. This again indicates a strong need for a single interface which allow users to issue querys to any number of available distributed information sources (i.e. networked queriable search collection) simultaneously.

In our system we have used indexes of multiple distributed collections to get broad coverage and by integrating the advantageous features of different presentation techniques (like ranking,clustering etc), try to present the result in a better manner, so that user can figure out the relevant documents with less effort.

1.2 The Scope of Our Work

In this section we are going to describe the scope of the problem, that is attempted in this thesis. The work in this thesis is aimed at designing and implementing a search tool which uses a meta search architecture for retrieving results and presents the result to the user by integrating techniques like, ranking classification and clustering.

Our design of search tool requires following features:-

- **Integration of Documents Presentation Technique** It is envisaged that

search engines results can be greatly improved by integrating techniques, like, *ranking*, *classification* and *clustering*, in the mechanism of the information presentation. Our aim is to use these presentation scheme and build a system which present retrieved result in a manner easily browsable to the user.

- **Superior Ranking Technique** We try to provide a better ranking scheme which should be capable of *prioritizing* the retrieved documents according to their relevancy.

- **Better Classification Scheme**: Our classification scheme should be able to classify the relevant and general pages from the retrieved set.

- **Decent Clustering Algorithm**: The clustering algorithm should be able to properly group the related documents together.

- **Maximum Coverage** Due to the tremendous growth in the web data in recent years, it is impossible for a single search engine to index all the web documents. So the one of the main challanges faced today by the Internet information retrival systems are incomplete search. Therefore our system should be able to provide maximum coverage.

- **Efficiently Browsable** The system should be able to present the result in such a fashion, which helps user in locating relevant information.

- **Scalability** The architecture of the system should be scalable, so that we can add any number of search engines inside it for incresing search coverage.

1.3 Organization of the Report

Rest of the thesis is organized as follows. Chapter 2 introduces the terms and concept of search engines and about their presentation technique which include ranking, classfication and clustering. Chapter 3 disscusses about the design of *KhojYantra*. The implementation is disscussed in detail in chapter 4. Experiment results and the future work is described in chapter 5.

Chapter 2

Background and Related Work

2.1 Background

Imagine you are searching for information in the world's largest library, where the books and journals(stripped of their covers and title pages) are shelved in no particular order, and without reference to a central catalog. It is virtually impossible for any one to search for a particular reference. Similarly the World Wide Web is like a virtual library. Instead of a central catalog, the web offers the choice of dozens of different search tools, each with its own database, command language, search capabilities, and method of displaying results.

2.1.1 Search Tools

Internet search tools are the tools that people use to find what they are looking for on the Internet, or facilitate their own sites being found on the web.

These tools are broadly classified into two categories.

- *Search Engines*

These search tools allow user to enter keywords that are run against a database (most often created automatically, by "spiders" or "robots"). Based on a combination of criteria (established by the user and/or the search engine), the search engine retrieves WWW documents from its databases that match the keywords entered by

the searcher. It is important to note that when you are using a search engine you are not searching the Internet "live", as it exists at this very moment. Rather, you are searching a fixed databases that has been compiled some time previous to your search.

While all search engines are intended to perform the same task, each goes about this task in a different way, which leads to sometimes amazingly different results. Factors that influence results include the size of the database, the frequency of updating, and the search capabilities. Search engines also differ in their search speed, the design of the search interface, the way in which they display results, and the amount of help they offer.

In most cases, search engines are best used to locate a specific piece of information, such as a known document, an image, or a computer program, rather than a general subject.

Examples of search engine include:
AltaVista:*http://www.altavista.com*
Google:*http://www.google.com*
The immense growth in online information during last few years results in development of variants of search engines like Meta-Search engine, Subject-specific search engines.

- **Meta-Search Engine**

  As it is impossible for single database to index all the documents, a secondary search engine, often referred to as multi-threaded search engines, has been developed which allow user to search multiple databases simultaneously, via a single interface.The meta search engine provide ability to sort the result by site, by domain and the ability to select which search engines to include. These modifications have greatly increased the effectiveness and utility of the meta search engines.

  Examples of meta search engines include:
  Meta-Crawler: *http://www.go2net.com/search.html*
  Dogpile: *http://www.dogpile.com*

- **Subject-specific Search Engine**

These search engines do not attempt to index the entire Web. Instead, they focus on searching for web sites or pages within a defined subject area, geographical area, or type of resources.Because these specialized search engines aim for depth of coverage within a single area, rather than breadth of coverage across subjects, they are often able to index documents that are not included even in the largest search engine databases. For this reason, they offer a useful starting point for certain searches.

Examples of Subject-specific Search Engine:
People(E-mail addresses)
Bigfoot: *http://bigfoot.com*
Music
The Ultimate Band List *http://www.ubl.com*

Subject Guides

Subject guides are hierarchically organized indexes of subject categories that allow the web searcher to browse through lists of web sites by subject in search of relevant information. They are compiled and maintained by humans and many include a search engine for searching their own database.They are best to searching for information about a general subject, rather than for a specific piece of information.
Examples of subject guides include:
Yahoo: *http://www.yahoo.com*
LookSmart: *http://www.looksmart.com*

The main goal of the Information Retrieval Systems is to lead the user to those documents that will best enable him/her to satisfy his/her need for information.To achieve these goals information retrieval community has developed some useful techniques of information presentation. Some of the techniques are given below.

2.1.2 Ranking

Ranking is the technique used by the Information Retrieval system to sort their results, according to some criteria, before presenting to the users. Criteria can include the number of terms matched, proximity of terms, location of terms within the document, frequency of terms (both within document and within the entire database), document length, and other factors. The exact "formula" for how these criteria are applied is the "ranking algorithm" and varies among search engines.More recently, several search engines have begun considering factors such as the number of links made to a page or the number of times a page is accessed from a results list in their ranking criteria.

2.1.3 Classification

Classification is a process of assigning incoming documents to one of some initial groups by using some predefined criteria. This process is essentially similar to filtering. These criteria may contain the similarity of document with the group, satisfying certain rules to get into the groups etc.

2.1.4 Clustering

In contrast to classification, document clustering involves discovering a set of categories to which documents should be assigned. Thus, rather than trying to simply assign documents to pre-defined categories, clustering algorithms are required to discover distinct categories. Documents in the dataset are then assigned (often as a by-product of the clustering process) to these newly discovered categories. To define more formally the notion of document clustering, consider a set D of m documents, denoted $d_1, ..., d_m$. We can partition D into K clusters, $c_1, ..., c_k$. The clustering task thus has two important components. The first is determining how many clusters to partition the data into (i.e., choosing K). The second is how to assign each document to these respective clusters.

Document Clustering has been extensively investigated as a methodology for improving document search and retrieval. The basis for using document clustering in

IR is the Cluster Hypothesis of van Rijsbergen : *closely associated documents tend to be relevant to the same request.*The idea is that the relevant documents are more similar to each other than to non relevant documents. If this hypothesis holds on a particular collection then it would make the retrieval more effective, because the class once found will contain only the relevant documents.Recently, a technique named S-catter/Gather introduced document clustering as a document browsing method.Our work follows the same paradigm.

Clustering Algorithms

The two most widely applied clustering algorithm to text domains are *hierarchical agglomerative clustering* (HAC) and *iterative clustering techniques.* Both of these methods rely on the definition of a similarity score between pairs of documents. For the sake of generality, we will refer to this similarity score as $Sim(D_i, D_j)$.

- Hierarchical Agglomerative Clustering: The most common clustering method employed in the information retrieval community over the past decade is hierarchical agglomerative clustering (HAC). This family of methods begins by placing each document into a distinct cluster. Pairwise similarities between all such clusters are computed, and the two closest clusters are then merged into a new cluster. This process of computing pairwise similarities and merging the closest two clusters is repeatedly applied, generating a *dendogram* structure that contains only one cluster (encompassing all the data) at its root. For example, Let there are five documents a,b,c,d and e to cluster. We first measure the similarity between documents and then combine the documents having higher similarity measure. So at initial steps (a,b) have higher value, we combine these to form cluster. Then again pairwise similarities between all the clusters are computed. At this point the similarity measure of d and e came higher so they are clustered together. Similar process repeats and finally we get a cluster having each documents in it. A sample dendogram is shown in figure 2.1

  HAC clustering method require the definition of a similarity or distance function between the documents. Each document is typically represented as a weight attribute vector, with each word in the entire document collection being an

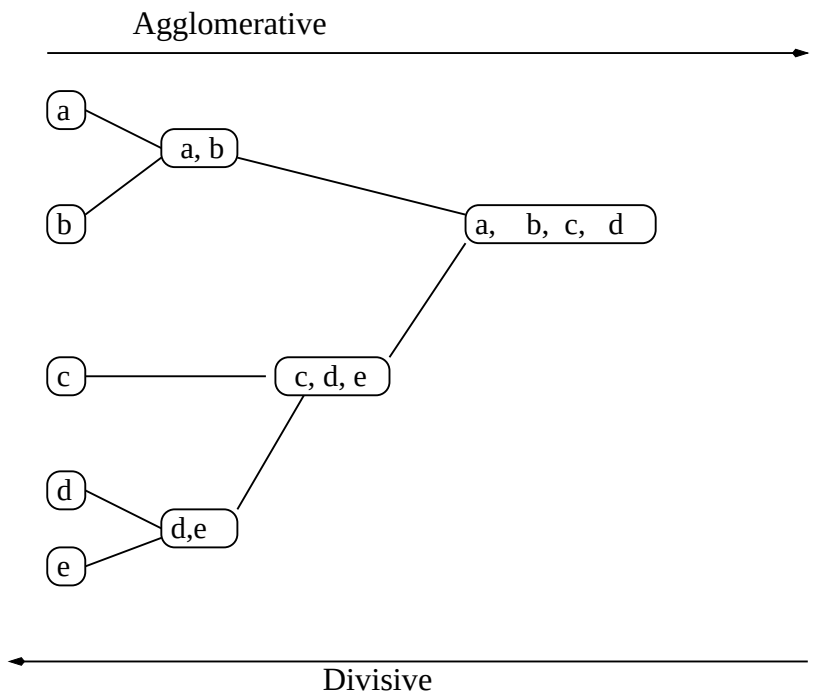


Figure 2.1: Example of a dendrogram

attribute in this vector. The *Cosine similarity* measure is commonly used as distance function in the information retrieval systems.

Depending on how the similarity of a document to a cluster is defined, we can obtain different "flavors" of HAC; the most common are the single link, complete link, and group average methods.

Single link: This is one of the simplest agglomerative hierarchical clustering method also often known as the nearest neighbor technique. This defines the similarity as the maximum similarity between any two individuals, one from each of the group. Although single -linkage clustering is known to have an undesirable chaining behavior, typically forming elongated straggly clusters, it remains popular due to its simplicity.

Complete links: The *complete linkage* or also known as furthest neighbor clustering method is the opposite of *single linkage* in the sense that they defines the similarity as the minimum similarity between any two individuals, one from each of the two group. The single link and the complete link measure is

illustrated in figure 2.2

Group Average linkage: In this method the similarity defines as the average similarity between all pairs of individuals that are made up of one individual from each group. A sample of Group Average Similarity measurement is shown in figure 2.3

Let the two clusters be C and C', the similarity measure between them calculated by different methods are as follows:

$$SLSim(C, C') = max(Sim(doc, doc')|d \in Candd' \in C') \tag{1}$$

$$CLSim(C, C') = min(Sim(doc, doc')|d \in Candd' \in C') \tag{2}$$

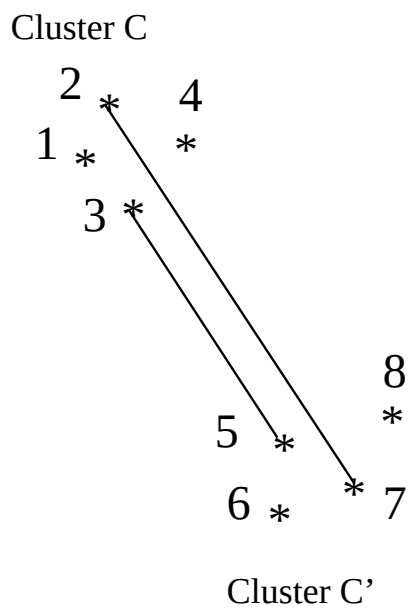
$$GALSim(C, C') = \sum doc \in C, doc' \in C' \frac{1}{|C|.|C'|} Sim(doc, doc') \tag{3}$$

$$= \sum doc \in C \frac{1}{C} Sim(doc, C') \tag{4}$$

where $SLSim(C, C')$ denotes similarity measure by single link, $CLSim(C, C')$ denotes similarity measure by complete link and $GALSim(C, C')$ is the similarity measure by group average link. doc and doc' are the documents from Cluster C and C'.

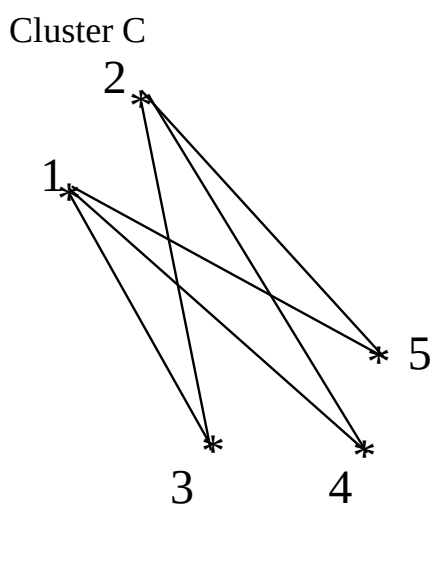
HAC algorithms are typically slow when applied to large document collections. These methods typically take at-least (n^2) time.

- Iterative Clustering: Iterative clustering techniques, also referred to as reallocation methods, attempt to optimize a given clustering by repeatedly reassigning documents to the cluster to which they are most similar. Let us suppose you have n documents to clusters. The number of final clusters are known previously, let that is K. This algorithms initiates by dividing the whole set of n documents into K sets and then calculate the similarity measure of each documents to the K sets and assign the documents to the set with whome its similarity measure is higher. The general form for such algorithms, given a specification of the number of clusters k, is as follows:



SLSim(C,C')=d35 and CLSim(C,C')=d27

Figure 2.2: Single linkage and Complete linkage distances



GALSim(C,C') = (d13+d14+d15+d23+d24+d25)/6

Figure 2.3: Group average distance

1. *Initialize the k clusters.*
2. *For each document doc in the corpus*
   (a) *Compute the similarity of doc to each cluster*
3. *For each document doc in the corpus*
   (a) *Assign doc to the cluster most similar to it*
4. *Goto 2, unless some convergence criterion is satisfied*

As in the case of HAC, we similarity of a document to a cluster by the any similarity method defined above.These algorithms typically have rectangular running times, i.e. $O(kn)$.

2.2 Document Representation

In the environment of information retrieval items to be processed by the computer are not simply "records" or "tuples" as in conventional database management systems.Instead, we have a collection of documents "Snippets". So we need to represent the meaning of text in a form more amenable to processing by computer.

One well known representation for the types of classification and clustering problems we consider,is vector space representation, described in detail in 2.2.1 .In this representation, documents are cast as vectors in a very high-dimensional space. Since clustering algorithms in vector space models can be computationally expensive with such high dimensionality, we examine some initial methods for dimensionality reduction in 2.2.2. Figure 2.4 shows the progression of stages involved as text documents are processed into vector representation with reduced dimensionality.

2.2.1 Defining a Vector Space

for documents commonly employed for many information access problems. In this representation, each document is characterized by a Boolean or numerical vector. These vectors are embedded in a space in which each dimension corresponds to a distinct term in the corpus of documents being characterized. A given document

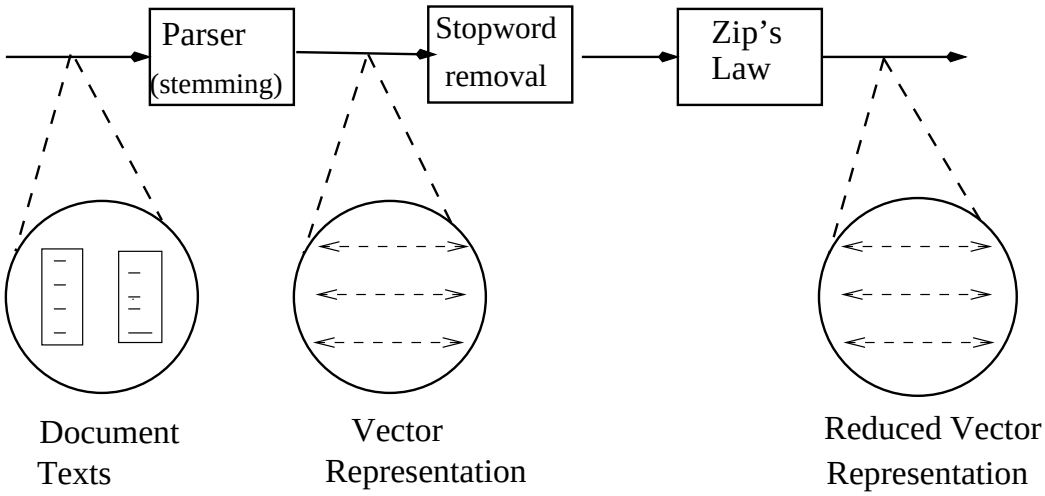


Figure 2.4: Initial Document Processing and Representation

vector has in each component a numerical value denoting some function f of how often the term corresponding to that dimension appears in the document. By varying the function f , we can produce alternative term "weightings". We explore some of standard weighting functions below, and give examples of vector representations of documents using these schemes.

Defining "Terms"

Although we have mentioned that the dimensions of our vector space correspond to distinct terms in the corpus of documents being represented, we have not defined what is to be considered a "term." The most common definition of a term (in English, as well as most other languages that use the Roman alphabet), and the one that we use in all subsequent chapters, is that a term is a sequence of alpha-numeric characters which is delimited by white space (spaces, tabs or newline characters) or punctuation marks (such as a period or a comma). Moreover, all uppercase letters in a document are converted to lowercase, so effectively capitalization is ignored. Consider the two short sample documents below

Table 2.3 shows the results of parsing these two documents into single-word terms, and then representing them as vectors with simple term frequencies (i.e. term counts) in each component. Such a representation is sometimes also referred to as a bag of words, since the relative position of terms in the document, and hence the language

Computing is not about computers any more. It is about living

Table 2.1: Sample Document No.1

To live is to compute!

Table 2.2: Sample Document No.2

structure, is not captured in the resulting vectors.

Terms	*vector for Document No.1*	*vector for Document No.2*
about	2	0
any	1	0
compute	0	1
computers	1	0
computing	1	0
is	2	1
it	1	0
live	0	1
living	1	0
more	1	0
not	1	0
to	0	2

Table 2.3: A simple vector representation of the sample documents

Word Stemming

In some cases, rather than defining terms to be the distinct words in the corpus, word stemming is used to reduce words to some root form. Thus, the terms that define the dimensions of the vector space are not actual words, but word stems. For examples, the words like *computer*, *computers* and *computing* would all reduced to word *compute*. We have used a stemming algorithm for this purpose, which has developed by Porter [18].

The stemmed version of the documents are as :

Compute is not about comput ani more. It is about live.

Table 2.4: Sample Document No.1

To live is to comput!.

Table 2.5: Sample Document No.2

Terms	*vector for Document No.1*	*vector for Document No.2*
about	2	0
any	1	1
comput	2	1
is	2	1
it	1	0
live	1	1
more	1	0
not	1	0
to	0	2

Table 2.6: A vector representation of the stemmed versionof the sample documents

Frequency-based Vector

As mentioned previously, in a vector space representation of documents, various functions may be applied to the frequency of term occurrences in documents in order to produce "weighted" document vectors.More formally, let ,$\xi(t_i, d)$ denote the number of occurrences of term t_i in document d.We may then apply some function f to $\xi(t_i, d)$ to produce the value for the ith component of the vector for document d. For the vectors in Table 2.3, for example, we simply used the identity function $f(\alpha) = \alpha$ applied to the term counts.

The most well-known function applied to document term frequencies is TFIDF weighting.This scheme, not only uses the term frequencies (TF) in each document used as part of the weighting function, but also consider the inverse document frequency

(IDF) of each term in the entire collection.More formally, IDF is usually defined as

$$IDF(t) = log(N/n_t) \tag{5}$$

where N is the total number of documents in the collection and nt is the number of documents in which term t appears at least once. The TFIDF weight for a term t in a document d is calculated by taking the product of the term frequency and the inverse documents frequency for that term,

$$TFIDF(t,d) = \xi(t,d) * IDF(t) \tag{6}$$

Instead of this function their are several other functions which are commonly applied to term frequencies like:

$$f(\alpha) = log(\alpha + 1) \tag{7}$$

which was defined by Robertson and Sparck Jones and successfully used for retrieval;

$$f(\alpha) = \sqrt{\alpha} \tag{8}$$

was used in the Scatter/Gather system [4] for document clustering and was found t outperform 6 for that task;

$$f(\alpha) = \frac{\alpha}{\alpha + const} \tag{9}$$

was also found to be useful for document retrieval (using various instantiations of the constant value).

Boolean Vector

Alternatively,we can also use simple Boolean representation of documents, in which we simply record whether or not a given term appears in a document.In this case function applied to term frequencies are like:

$$f(\alpha) = \begin{cases} 1 & if \quad \alpha \geq 1 \\ 0 & otherwise \end{cases} \tag{10}$$

The underlying Boolean model is generally used by various classification methods where they need only the information of presence and absence of a particular word in documents.

2.2.2 Controlling Dimensionality

In using a vector space representation for documents, it becomes clear that the resulting dimensionality of the space will be enormous, since the number of dimensions is determined by the number of distinct terms in the corpus. For example, feature spaces on the order of 10^3 to 10^5 are not uncommon for even reasonably small collections of documents. The problem of high dimensionality is further exacerbated in very heterogeneous environments, such as the World Wide Web.

about	day	if	thru
above	did	is	too
alone	differ	itself	same
along	do	just	see
also	each	km	than
am	else	let	the
as	enough	like	to
at	etc	may	toward
be	ever	neither	unable
became	far	often	unless
because	farther	others	upon
behind	few	ought	various
came	for	per	want
can	gave	provide	we
canst	go	quit	what
certain	has	rather	why
clear	have	round	yet
click	her	said	you

Table 2.7: Sample List of StopWords

Removal of StopWords

The words like prepositions, conjunctions and pronouns are used to provide structure in language rather than content. Such words are commonly used in documents regardless of topic, and thus does not contain any useful information. As they will of little use when clustering or classifying documents, we can eliminate such words (and the dimensions corresponding to them) from our document vectors. These words are

commonly referred as stop words. A list of exemplary stop words is given in Table 2.7.

Currently, in KhojYantra we have a stopword list of approximately 470 common English words and an additional 20 commonly used words on the Web.This process helps in removal of several hundred dimensions from our vector space.As the dimensionality of the space is very large we need some additional elimination.

- *Zip's Law*

To further reduce the dimensionality of the document vectors, we make use of another well known phenomenon: many words in a corpus appear very infrequently. As our goal is to identify similarities and differences among an entire collection of documents, then words which only appear, say, once or twice (or generally infrequently) in the collection will have little resolving power between documents [22].

The justification for the elimination of such infrequent terms lies in an observation about the frequency of word appearances in collection made by Zipf over 50 years ago [7]. Since that time, this observation has been named "Zipf's Law," although it is not actually a law, but merely an empirical and approximate mathematical phenomenon.

2.2.3 Related Work

To better understand the technical contribution encompassed in the development of *K*hojYantra, it is important to get an overview of related work in the area of search engines.

Currently on the net, different techniques adopted by the search engines to help the user in making his search according to his preferences and in sifting large set of retrieved results. The details of the some major search engines and their techniques are described as follows:

AltaVista: AltaVista is consistently one of the largest search engines on the web, in terms of pages indexed. Its comprehensive coverage and wide range of power searching commands makes it a particular favorite among researchers.AltaVista opened in December 1995. It was owned by Digital, then run by Compaq (which purchased Digital in 1998), then spun off into a separate company which is now controlled by

CMGI.It offers the option to view more pages from each site by doing site based clustering [2].

Excite: Excite is one of the most popular search services on the web. It offers a medium-sized index and integrates non-web material such as company information and sports scores into its results, when appropriate. Excite was launched in late 1995.It provides "More Like This" option which help user in identifying the similar documents. This is helpful when you find relevant hits and want to see more on the same aspect of the topic.It also provides the "List by Web Site" option that helps users to locate the documents from a particular site.[6]

Northern Light: Northern Light opened to general use in August 1997. It tries to do concept based clustering of its documents and organizes results into Custom Search Folders.These folders can be based on source(e.g. commercial Web sites,domains etc),type(e.g., press releases, product reviews etc),subject(e.g.,hypertension, baseball etc.) or language(e.g., English, French etc). NorthernLight does pre-clustering of its pages by assigning, at the time of indexing, a set of potential folders to each document. It then decides which folders to present at the query time.[17]

Yahoo: Yahoo is the web's most popular search service and has a well-deserved reputation for helping people find information easily. It provides a hierarchical subject organization that is good for browsing.[25]

Google Google is a search engine that makes heavy use of link popularity as a primary way to rank web sites. In determining relevancy ranking, the engine also looks at various textual clues including linking text. Engine does not stem words; it searches on your word form exactly as it is typed.[8]

Grouper In the recent past years a new technique has been developed, which presents the result after doing dynamic document clustering of online search outputs. This search engine uses the similar post clustering concept in its result presentation.[9]

Chapter 3

KhojYantra - Design

In this chapter we are going to describe the design of our system, in which we have tried to overcome the drawbacks of conventional search engines and provide the features which help the user in finding relevant information. In the first two section we mention our design goals and describe the architecture of our system. Then we look at the various parts of our architecture and their functions. Details of the implementation of each part will be discussed in the next chapter.

3.1 Design Goals

The design goals of *KhojYantra* are as follows:

1. **Integration of Document Presentation Techniques**: Instead of attempting to reduce the number of results returned by the search engines (e.g., by filtering methods or by advanced pruning options), our aim is to make the retrieved results easy to browse. It is envisaged that the search results can be greatly improved by integrating techniques, like *ranking*, *classification* and *clustering*, in the mechanism of the information presentation.

2. **Superior Ranking Technique**: Current search engine uses very simple ranking algorithm,which results in scattered presentation of relevant documents in their results. So one of the prominent feature of our design is that our ranking

technique should be capable of *prioritizing* the retrieved documents according to their relevancy.

3. **Better Classification Scheme**: Our classification scheme should be able to classify the entire set of retrieved documents into most relevant, relevant and general groups. The *most relevant group* contains the pages from the web server mainly devoted for the information related to the search terms, *relevant pages* are the pages that contains a lot of information related to the query and the *general group* is supposed to be a collection of documents less relevant or not at all relevant to the given query. The justification of this approach is described in section 3.4;

4. **Decent Clustering Algorithms**: The clustering algorithm should properly group the related documents together. The documents may be related in any manner (e.g, in their meaning, in their origin etc.). It has been shown that post-retrieval document clustering produce superior results than pre-retrieval clustering [26]. This is due to the fact that clusters are computed based on the returned document set; the cluster boundaries are drawn to appropriately partition the set of documents at hand. In contrast, pre-retrieval clusters might be based on features that are infrequent in the retrieved set, as many non-retrieved documents influence the cluster formation. So in our design we have chosen the post clustering technique, which, as discussed above, will result in a good quality clusters.

   (a) **No Preprocessing**: Since our clustering is applied to a dynamically generated set of documents, it cannot rely on preprocessing of the documents to improve its efficiency.

   (b) **Snippet-Capable**: The algorithm should produce "reasonable" clusters even when it only has access to the short document snippets returned by the search engines.

5. **Maximum Coverage** Due to the tremendous growth in the web data in recent years, it is impossible for a single search engine to index all the web documents. So the one of the main challenges faced today by the Internet information

retrieval systems are incomplete search. Therefore our system should be able to provide maximum coverage.

6. **Ease-of-Browsing**: The result should be easily browsable i.e., user should able to determine at a glance whether the contents of documents or clusters are of his/her interest or not.

7. **Scalability**: The architecture should be scalable, so that we can add any no of search engines inside it for querying.

3.2 Architecture of KhojYantra

The complete process of the system can be broken down into following steps:

1. Take the user's query, convert it into appropriate format and send it to each specified search engine. Construct the initial result set by collecting the results returned by each search engine.

2. Remove the duplicate documents from the initial set.

3. Classification of the reduced result set into *relevant groups* (Most relevant group and relevant group) and *general group* based on presence of the search term in the url text.

4. Process relevant document groups and general document group in parallel.

5. Present the result to the user

In the following sections, we will examine each component of the system in detail.

3.3 User Interface

We provide a interface for the user in which he can specify various parameters, like how query terms are related(e.g., as Any Word, All Word etc.), which search engines to run, timeout value, number of hits to be retrieved from each of the selected search

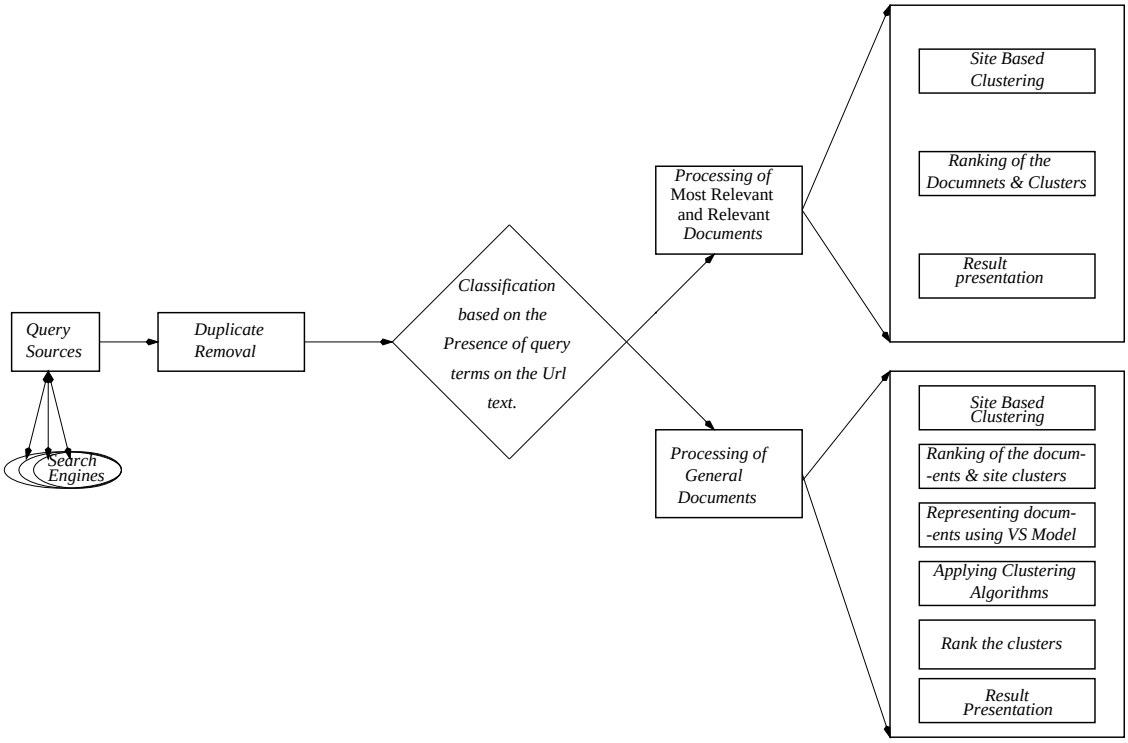


Figure 3.1: The Architecture of KhojYantra

engines etc, to make his search according to his preferences. We have also set some default values for all parameters to help the casual users. Details are discussed in next Chapter.

3.4 Obtaining initial set of Documents

The design of the system is based on the ***multi-engine search architecture***.Under this architecture, the system receives a user's query as a list of keywords, converts it into acceptable format pertaining to each of the search engines selected by the user and fire it in parallel. Search engines try to retrieve documents till they get timed-out. Thus, we get the initial set of documents by collecting all the results returned by the search engines . The reason of choosing this architecture instead of single standard search engine is that standard search engines have limited coverage, outdated databases, and are sometimes unavailable due to problems with the network or the engine itself. By choosing this architecture we attempt to contend with the

problem of limited coverage. The primary advantages of this architecture is that we can combine the results of several search engines and present a consistent interface to the user.

3.5 Documents Processing

As various search engines overlaps in their coverage, there is a very high probability of getting similar urls from different search engines. Moreover due to host name aliases, symbolic links, redirection and other forms of obfuscation we get the references of same document in different forms. In the Documents Processing Phase we detect similar pages and remove them from the Initial Document Set.

3.5.1 Duplicate Removal

Detecting duplicate references is difficult without the full contents of a particular page. In *KhojYantra* we handle this problem with the help of simple heuristics developed after analyzing the naming conventions of the documents on the web. Our heuristics consider the cases like
URL of the form
http://www.cse.iitk.ac.in/rajeevm/
often refers to
http://www.cse.iitk.ac.in/rajeevm/index.html.
Similarly
http://www.cesoft.com/default.html
refers to
http://www.cesoft.com/
and
http://google.com
refers to
http://www.google.com.
We check for all combination of the above cases and remove the duplicate documents from our initial set.

3.6 Classification of Initial Set of Documents

In this phase, we are trying to classify the initial set of documents into three groups, namely *most relevant*, *relevant* and *general* set, based on the presence of the search words in the url of document. If any of the search words are present in the site name then we put such documents in most relevant group else if any of the search words are present in the rest of the url then we put it in the relevant group. Otherwise we put the document in the general group.

3.6.1 Intuitive Justification of this Approach

The idea behind using this approach is mainly based on the concept that if any of the search term is present in the site name then that site contains good information regarding that term and most relevant for that search word. For example, if the search word is "java" the site like
http://java.sun.com,
http://javaworld.com
are most relevant.similarly if the query word is "bsd linux" the relevant sites are like
http::://bsdlinux.lgg.ru/
http://www.netbsd.org/.
Based on the similar concept, if any of search word is present in url text other than site name then also that document contains a valuable information related to the term. For example, for word "java", sites like
http://www.ibm.com/java/
or
http://www.cse.iitk.ac.in/users/rajeevm/javaresouces.html
contains good and relevant information related to java.

And finally we consider all other documents as *general* documents. Among these documents many are either less relevant or not at all related to the search terms. The reason of getting not relevant pages are due to the simple ranking algorithms used by the search engines. These ranking can be further muddled by the keywords "spamming" to increase the page's rank order.

After classifying the documents in groups, we apply different algorithms for processing the documents in each group.

3.6.2 Processing of documents in the *Relevant Groups*

The processing of documents in this group is done in two phases. In first phase we do the clustering of documents based on their similarity of origin and in the second phase, we measure the relevancy of each documents. These phases are described as follows:

- *Site based Clustering*

Site based clustering is done by developing the cluster containing the documents which are from the same site. We represent each cluster by the site address and with the title of two top ranked document in the cluster.

- *Ranking of Documents and Clusters*

After doing the site based clustering we calculate the ranks of each document present in the set by measuring its relevancy with respect to search terms. As in this set the search terms are present in the urls, the relevancy of the document is totally dominated by the *position*, *proximity* and the *presence* of the search terms in the url text. Hence we consider only the url text during the relevancy measurement of the documents in this set. The ranking of the clusters is measured by calculating average rank of all the documents present in the cluster.

3.6.3 Processing of Documents in the General Group

As we mentioned in the previous section the documents that form part of this group, are less relevant to the search words; hence we need to present these documents into some small predefined number of clusters. The processing of documents in this group includes the following steps:

Clustering Based on Site Address

In this phase, we do site based clustering similar to what we have done previously in the processing of relevant documents.

Ranking of Documents and Clusters

In this phase we take the document set and again use the ranking algorithm to calculate the relevancy of the document with respect to search terms. The difference between this algorithm and the previous one is that instead of using url text this one uses Title text and Abstract text for relevancy measurement. As there is no search terms present in the url text there is no need to consider it. We normalize the ranks of cluster by taking the average of rank of documents present in the cluster. The algorithm is described in detail in the next Chapter.

Obtaining Data Matrix

Due to the simplicity and popularity of *vector space model* in the information retrieval systems we have chosen it for our document representation. Each document in the set is represented by numerical vector in the multidimensional space where each dimension corresponds to a distinct term in the corpus of documents. To describe it more formally, Let us suppose for each document α in a corpus C, *c(α)* be the set of words with their frequencies, that occur in that document. Let W be the set of unique words occurring in C. Then *c(α)* can be represented as a vector of length $|W|$

$$c(\alpha) = f(w_i, \alpha)_{i=1}^{|W|} \tag{1}$$

where w_i is the ith word in W and $f(w_i, \alpha)$ is the frequency of w_i in α.

We then apply *stemming algorithm*, *stop word removal* process and *zipf's law* in sequence to reduce the dimensionality of vectors. After this we get a reduce vector for each document. We than represent the entire set by (documents X terms matrix), where each row represents a document and each column represents a term present

in the corpus of documents. As the ratio of terms present in the document with the terms present in the whole corpus is very very small, we get a matrix sparse in nature, as a result. Each entry(i,j) in the matrix represent the weight of that jth term in the ith document.

- *Clustering Algorithms*

In this phase we are trying to cluster the documents based on the similarity of their "meaning". We have already discussed in previous chapter about the advantages and disadvantages of hierarchical and partitioning algorithms, which are traditionally used by the information retrieval community for clustering. We have chosen an algorithm which is initially proposed in[4], which use techniques drawn from hierarchical algorithms, but which achieve rectangular time bounds of partitioning algorithms. For our application, as the number of desired clusters are small thus the speed of our algorithm is much better than quadratic time algorithms. The details of the algorithm are discussed in next chapter.

- *Ranking of Clusters*

After getting the cluster from the previous phase, We sort the cluster according to their relevancy with the search terms. We calculate the rank of each cluster by taking the average rank of all the documents present in the cluster. These cluster contain the documents as well as the cluster of the documents from the same site.

3.6.4 Results Presentation

The final phase of our design is the presentation of result to the users. We have designed our interface keeping in the mind that in results presented should be easily browsable by user,i.e. user will able to determine the relevant documents or document clusters at a glance . The main design decisions have been taken in KhojYantra are

- *Presentation of Classified Groups in Browser Window*: The issue here is how to present the three constructed group of documents to the user so that he can get the interested document easily. We have presented the result by dividing the

browser window in three parts for each group. This approach helps in giving the overall view of the whole result to the user. As user is the better judge of the relevancy of the documents, the top relevant documents and clusters in each group are shown simultaneously so that he/she can get the interested documents at a glance.

- *Cluster Representation*: The decision issue here is how to represent the cluster in the result so that it must be sufficient enough to reveal the user a sense of topic defined by the documents inside it. It must also short enough so that many descriptions can be appreciated simultaneously.We have adopted two different schemes for representing clusters present in the relevant groups and general group.

- *Ordering of Documents inside a Cluster*: The issue here is how to order the documents in a cluster. This effects both the order of the documents on the cluster's page, and the cluster's summary on the main result page. Two options were considered : sorting the documents based on their relevance to the query or sorting them by nearness to the cluster centroid . We have chosen the first approach as ranking within a cluster by nearness to a query performs better than ranking by nearness to the cluster centroid [10] .

Chapter 4

Implementation

In this chapter, we are going to describe the implementation of our meta-search engine *K*hojYantra. The implementation includes the development of several modules for handling various steps in the complete cycle of result retrieval. The implementation as described in this chapter is commensurate with the design decision taken in the previous chapter.

4.1 User Interface

A KhojYantra session starts with the user entering her query in the query box. There are several options present in the interface which user can use to make his search more accurately and according to his/her requirements. The user can choose how query terms are related(e.g., as Any Word, All Word etc.), which search engines to run, timeout value, no. of hits to be retrieved(10-100)from each of the selected search engines. Our engine supports all common search formats, including Boolean Syntax. As many other meta-search engines, khojYantra dynamically modifies queries to match each search engine's query syntax.

4.2 Multi-Engine Search Architecture

We are currently using eight search engines for querying.These are *AltaVista*, *Google*, *AlltheWeb*, *Excite*, *HotBot*, *Yahoo*, *InfoSeek* and *WebCrawler*. The choice of these

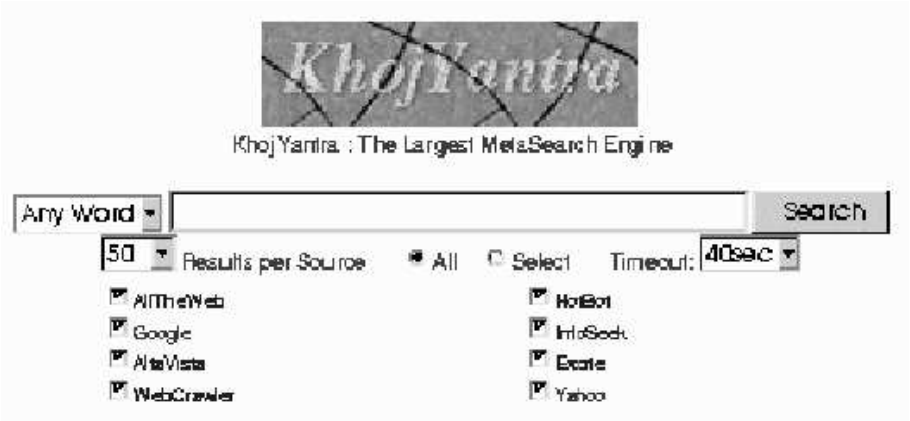


Figure 4.1: KhojYantra's Interface

search engines are basically influenced by popularity,high coverage and the less overlapping indexes [21]

We have implemented the different module for retrieving and parsing the search results, of each search engines. we have used a web scripting language called WebL(pronounced "webble") [24] for building these modules. The advantage of using this over other general purpose languages is that it is specifically designed for the retrieving documents from the web, extracting information from the retrieved documents, and manipulating the contents of documents.

4.3 Document Processing and Classification of Initial Set

Now after having retrieved the results from all or specified search engines, We are going to remove the duplicate documents from the set. As it is quite difficult to identify the similar documents without reading the full text. We have adopt some general heuristics, as discussed in previous chapter, for identifying them. Our structure of storing search results or as follows,

Title: Stores the title of the page

Abstract: Contains the lines except than title and url present in the *snippet*

Url: Store the url text of the page

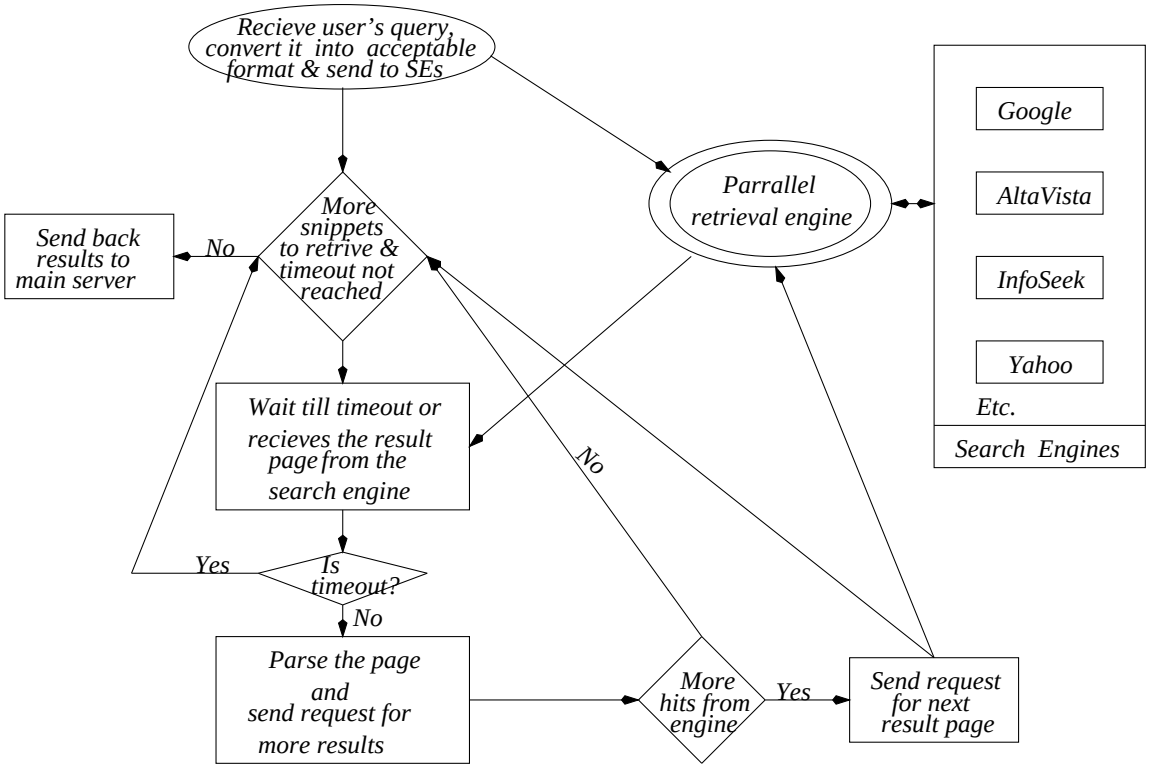


Figure 4.2: Multi-Engine Search Architecture

SearchEngine Name: Store the name of the search engine retrieve the page

After identifying the similar documents in the set , We remove all duplicates from the set and add their searchEngine Name in the SearchEngine Name of their representative. At the time of result presentation we give the name of all search engines, from which this document came, to the user, so that they can see the popularity of the page.

Now, we have in hand the reduced document set. We then apply our classification method to filter the most relevant pages from the set. For this purpose we have developed a algorithm which is described as :

1. Take a document.

2. Put the document in most relevant group, if any of the search terms are present in Site Address of the Url.

3. Otherwise, Put the document in relevant group, if any of the search terms are present in the Url text other than the site address.

4. Otherwise, Put the document in general group.

Now, at this point we have three groups in hand. Two relevant groups and one general group. We are handling the processing of relevant groups and the general group differently in our system.

4.4 Processing of Relevant Document Set

As these documents are highly relevant to query and contains a lot of information related to search terms.We have adopted a different criteria for their presentation. As the only problem in these type of results are documents from similar site are scattering here and there in the whole result, and user have to sift all the results if he is interested in the results from the same site. We achieved an additional advantage of better organization and presentation of results by doing this.

So to remove this problem, we have chosen to do site based clustering of the documents.

4.4.1 SiteBased Clustering

In the site based clustering we make clusters of the documents which are from the same site. For implementing this we have used the *HashTable*.The site address of the document is used as a key of the Hashtable. Therefore the Pages from same site are hashed on the same bucket. For example the urls like, *http://searchenginewatch.com/sereport /current.html* and http://searchenginewatch.com/facts/ataglance.html are mapped to the same bucket having key value *searchenginewatch.com*. After doing the site based clustering, we get the result set consists of the clusters and documents.Now we need some method to represent these clusters. One option is to choose the site address as the cluster descriptor, but the problem in that user can not guess about the content of the pages inside the cluster by just seeing the site address. To get the idea he will have to go through all the clusters. So for helping the user in locating relevant information easily, we have chosen a method, described in next subsection, for cluster description.

- *Cluster Descriptions*

We represent each cluster by the site address followed by the no. of documents inside the cluster. Then for the ease of the user we attempt to convey the contents of the documents inside the cluster by presenting the titles of the top two relevant pages of the cluster.The advantages of this representation is that it summarizes well about the cluster, so if cluster is of interest he/she can go to the cluster's page or if certain displayed title seems promising, the user can go directly to that documents by clicking on the title.

4.4.2 Ranking Algorithm

Now, we have all the documents and clusters are in hand. We have to calculate the relevancy of the documents and clusters with respect to query terms, So that we can put more relevant pages higher at the time of result presentation. The relevancy measure is computed by considering the *position of the first occurence of any of query terms* in the url text, *number of query terms present*, the *proximity between query terms* , and *the term frequency.* We have used the following equation for the documents containing more than one of the query terms :

$$Rank = (c_1 - firstAppereance) + c_2 * N_a + \frac{\left[c_3 - \frac{\sum_{i=1}^{N_a-1} \sum_{j=i+1}^{N_a} min(d(i,j), c_3)}{\sum_{k=1}^{N_a-1} (N_a - k)} \right]}{\frac{c_3}{c_2}} + N_t \quad (1)$$

where firstAppereance represent the minimum distance from the start of the Url text where any of the search terms occurs. N_a represent the no. of query terms that appear in the Url text(each term is counted only once); N_t is the total no. of query terms present in the url text. $d(i,j)$ is the minimum distance between the ith and jth query terms (currently measured in the no of characters); c_1 is the constant that controls the overall magnitude of Rank. c_2 is a constant that specifies the importance of presence of query terms. c_3 is a constant that specifies the maximum useful distance between the terms. Some context and page analysis for improving

web search results was also done [14]. The difference between the [14] and our algorithms is that we have consider the factor of firstAppereance of any of the search terms in the text during the calculation of relevancy measure. We have also given different importance weigth to each factors. For our implementation purpose we have chosen(c_1=2000,c_2=10,c_3=1000). For the case, where only one term is present we use the variant of the above formula which is as,

$$Rank = (c_1 - firstAppereance) + c_2 + N_t \tag{2}$$

The full url text is used for ranking of relevant documents, but we have chosen only the part of url text containing the site address for the ranking of most relevant documents.As we are only considering the presence of search words in the site address at the time of classification of most relevant pages.

We calculate the ranks of clusters by calculating the average rank of all the documents present in the cluster. Ties can be broken by assigning higher rank to the cluster containing higher no of documents inside it. The single documents are considered as clusters of size one.

4.4.3 Result Presentation

After calculating the relevance of each documents, only the presentation part is left to handle. We have used two frames to present the most relevant and the relevant result sets. We present the result in the frames in decreasing order of their relevancy. The sample window presents the result is shown in figure 4.3

4.5 Processing of General Document Set

As discussed earlier the documents in this set are less relevant or at worse case not at all relevant to the query. We have adopted a different approach for presenting these documents. We take the whole set and cluster it into some small number of sets based on their similarity before presenting to the user.The choice of the no. of cluster is totally influenced by the no. of documents to cluster. We describe each phase of processing in detail in the following subsections.

4.5.1 SiteBased Clustering

In this phase we apply the clustering criteria based on the similarity of site address, similar to as we have discussed in the sitebased clustering of relevant pages.

4.5.2 Ranking Algorithm

After clustering the result based on the site address, we calculate the ranks of all the documents and clusters. The ranking algorithm used in this case is variant of one discussed in the section of processing of relevant document set eqaution 1. In this case our algorithms is as follows,

$$Rank = c_1 * Rank_t + Rank_a; \tag{3}$$

where c_1 is the constant that controls the overall magnitude of Rank. $Rank_t$ is the magnitude of relevancy calculated by equation 1. We use Title text during calculation instead of Url text. Similarly $Rank_a$ is the relevancy calculated by the equation the following equation for documents containing more than one query terms in abstract text.

$$Rank_a = c_1 * N_a + \frac{\left[c_2 - \frac{\sum_{i=1}^{N_a-1} \sum_{j=i+1}^{N_a} min(d(i,j), c_2)}{\sum_{k=1}^{N_a-1} (N_a - k)} \right]}{\frac{c_2}{c_1}} + N_t \tag{4}$$

For our implementation purpose we have chosen(c_1=10,c_2=1000). The meaning of N_a and N_t are similar as described in equation 1. The variant of the equation 4 is used when only single query term is present in the abstract.

$$Rank_a = c_1 + N_t \tag{5}$$

As the abstract of documents "snippets" return by the search engines are chosen differently by the search engines for the same documents. We have not considered here the factor of first occurrence of search words.

Now at this stage we have all the documents with their relevancy value. We also calculate the rank of clusters in similar fashion as we have done previously by taking the average rank of all the documents.

4.5.3 Obtaining Data Matrix

After calculating the relevancy of each documents and clusters, now our main aim is to make the clusters by grouping the similar documents together. For this purpose we have to apply the clustering algorithms on the received document set. The input for the clustering algorithms is required in the form of ($documantXdocument$) matrix whose each entry (i,j) corresponds to similarity of ith document with the jth document. To build the matrix, we represent each document by using *vector space model* . In this model documents are cast as a vectors in a very high dimensional space .Since clustering algorithms in such a high dimensional space can be computationally expensive, we apply some initial methods (like, stemming, stopwords etc,), explained in chapter2, for dimensionality reduction.

For a given document vector d, the value of d_t,shows the importance of term t in document d, which is equal to $c * d_{1t} + d_{2t}$, where d_{1t} is the number of times the word appeared in title or url text, d_{2t} is the number of times the word appeared in abstract and as title and url text seems to carry additional contextual importance compared to the abstract we increase their weight by the factor c, known as importance factor. The weighting technique we are using in *KhojYantra* is $t$erm frequency weightingscheme, in which we use the identity function $f(\alpha) = \alpha$ i.e we replace d_t by itself. Generally it is common in *Information Retrieval* to weight term based on Inverse Document Frequency (IDF) in addition to term frequency, but we chose not to do so . The reason behind that is the preliminary studies done by [20] indicates IDF weighting slightly degraded cluster quality. IDF weighting down weights frequent words and up weights rare words, which may be useful for similarity search but complicates clustering since clusters tend to be formed based on patterns of frequent words.

As we have discussed earlier, the document vectors are quite sparse in nature. Therefore we have adopted a different representation for document vector as a vector of(term,weight) pairs; those terms are not listed have weight equal to 0. Thus a document with p non-zero weighted terms can be written as $P = ((y_1, d_1), ..., (y_p, d_p))$.

Now, the main bottleneck in clustering text documents is calculating the distance(similarity) between the document vectors. This calculation takes time linear in the document sizes. Thus there is need of some technique to reduce the size of

the document. One obvious way to do this,is to project each document onto a small subspace of the total term space, thereby reducing the average number of terms in each document. There are two different approaches to projecting documents. One is a *l*ocal method, where for each document we excise a number of "unimportant" terms. This type of projection, called *t*runcation, is called local because each document is projected onto a different subspace. The alternative to local projection is *g*lobal projection, in which the terms to delete are chosen first, and then these terms are deleted from each document. This type of projection is called *d*imension reduction. As the experiments done by [20] shows that there is no difference in clustering effectiveness between the projections. Now for the implementation purpose the main criteria is efficiency. The result of experiments also shows that TF-50, LSI-50 and LSI-150 are the most efficient projections, where LSI-d is LSI dimension reduction to d dimensions and TF-c is the dimension truncation to c dimensions. However, the LSI projections require an expensive "compile-time" calculation. Therefore we have decide to use moderate truncation, to about 50 terms per document, in the implementation of our clustering algorithm. Moderate truncation combines optimal clustering effectiveness and impressive time efficiency. [20]. For the purpose of truncation we have applied following rule. *if(no. of non-zero weighted terms in a document*> 50*) sort the terms in descending order of their weight and select the first 50 terms to represent the document.* We have also chosen the similar TF-50 projection for representing the cluster centroid. In practice, generally the cluster centroids are very dense, thus the application of truncation here, is very beneficial.

4.5.4 Similarity Measures

Now, we have in hand the truncated vector representation of each document. In order to cluster the documents one must first calculate pairwise measure of documents similarity and then define a method to partition the collection into clusters of similar documents.

The degree of similarity between document pairs is based on the weights of the corresponding matching terms. In our system cosine measure has been used for this purpose. Let D_i and D_j are two document vectors, the similarity measure between

them is calculated as

$$sim(D_i, D_j) = \frac{D_i \cdot D_j}{||D_i||||D_j||} \tag{6}$$

Thus similarity measure is computed in time linear in the document sizes. If the number of documents are K and having size w then the time to calculate the similarity between each pair of document is of $O(K^2w)$.

4.5.5 Clustering Algorithm

As we have discussed the advantages and disadvantages of hierarchical and flat clustering strategies in chapter 3. Here we have chosen to use a algorithm which uses the best of both strategies, uses techniques drawn from the hierarchical algorithms, and achieve rectangular time bounds similar to flat clustering algorithm. As in our case the number of cluster required is very small, thus speedup over quadratic time algorithms is substantial.

For defining the algorithm in more formal way. Let us assume the number of cluster required is k. So our algorithms starts with initial m centers, developed by breaking the whole corpus C into N/m buckets of a fixed size $m > k$. The cluster subroutine, discussed later in this section, is then applied in parallel to each of these buckets to agglomerate individuals into document groups. We assume the termination condition for subroutine loop as when the reduction in number (from individuals to groups in each bucket) is roughly a factor of σ. Now the resultant groups are treated as they were individuals and the entire process repeated. The iteration terminates when only k groups remain. This algorithm can be viewed as building a $1/\sigma$ branching tree bottom up, where the leaves are individual documents, terminating when only k roots remain.

Suppose the individuals in the corpus C are enumerated, so that C=$\alpha_1, \alpha_2, ...\alpha_n$.

The initial bucketing creates a partition as B = $\Upsilon_1, \Upsilon_2, ..., \Upsilon_{n/m}$, such that $\Upsilon_i = \alpha_{m(i-1)+1}$,$\alpha_{m(i-1)+2}$,...,$\alpha_{mi}$. Each Υ_1 is then separately clustered(using the cluster subroutine) into σm groups, where σ is the desired reduction factor. All these computation takes m^2 time, and, hence, all n/m takes nm time. Each application of agglomerative clustering produces an associated partition $R_i = \Phi_{i,1}, \Phi_{i,2}, ..., \Phi_{i,\sigma m}$. The union of the documents groups contained in these partitions are then treated as

individuals for the next iteration. That is, define
$C' = \Phi_{i,j} : 1 \leq i \leq n/m, 1 \leq j \leq \sigma m$
The process is then repeated with C' replacing C. That is, the σn components of C' are broken into σn/m buckets, which are further reduced to σ^2n groups through separate agglomeration. This process terminates at the iteration j if $\sigma^j < k$. At this point one final application of agglomerative clustering reduce the remaining groups to a partition P of Size k.

Now to determine the running time of this algorithm, observe the jth iteration, which operates on σ^jn items.This iteration takes σ^jnm time. So the overall running time is O(nm(1+σ+σ^2+...))=O(nm). Thus if m=O(k) this algorithm has rectangular running time.

As we have discussed the representation of cluster centroid is done by using TF-50 dimension reduction scheme. We represent the cluster centroid by taking atmost 50 most highly weighted entries i.e., keywords. Once we have found k centers and choose suitable representation for each cluster centroid , each document in C must be assigned to one of those centers based on some criteria. We use a simple algorithm, named *A*ssign-to-Nearest, to assigns each document to the nearest center. Let G be a partition of the collection into k groups, and let Θ_i be the ith group in G. Let $\alpha \in \Phi_i$ if i maximizes $sim(\alpha\Theta_i)$. Ties can be broken by assigning α to the group with lowest index. The set $P = \Theta_i, 0 \leq i \leq k$ is then the desired partition. Similar type of algorithmic approch is used in [4]. P can be efficiently computed by constructing an inverted map for the k centers, and for each $\alpha \in C$ simultaneously computing the similarity to all the centers. In any case, the cost of this procedure is proportional to kn.

Cluster Subroutine

As we have seen, in chapter 2, that trade-off between speed and clustering quality appears to be most satisfying for Group-Average method. We have chosen this method for measuring the distance between clusters.

The average similarity between document pairs in a document group say,Λ is

defined by

$$S(\Lambda) = \frac{1}{|\Lambda|(|\Lambda|-1)} \sum_{\alpha \in \Lambda} \sum_{\beta \neq \alpha} sim(\alpha, \beta) \tag{7}$$

Let G be a set of disjoint document groups. In each iteration of average agglomerative clustering we finds the two different clusters Θ and Δ which maximize $S(\Theta \cup \Delta)$ over all choices from G.

We merge these two groups(ie., Θ and Δ) and produced a new reduce partion G' as $G' = (G - \{\Theta, \Delta\}) \cup \{\Theta \cup \Delta\}$
Initially each group in G contains single document.So at the time of start the size of G is equal to the number of individual documents. The iteration stops when the size of the group is reduce to the number of clusters required. More formally when $|G| = k$. The final output from this procedure is the flat partition G', rather than a nested hierarchy of partitions, although we can compute the latter by recording each pairwise join as one level in dendrogram.

4.5.6 Result Presentation

As these documents are less relevant to the query words we planned to present it into some predefined number of clusters, so that user can get the overall view of the documents in this set. We represent each cluster by top ten keywords present in the cluster along with the title of top three pages inside the cluster. The representation helps user to get an overview of all the documents inside the cluster. User can also choose any of the presented title if it seems promising to him.

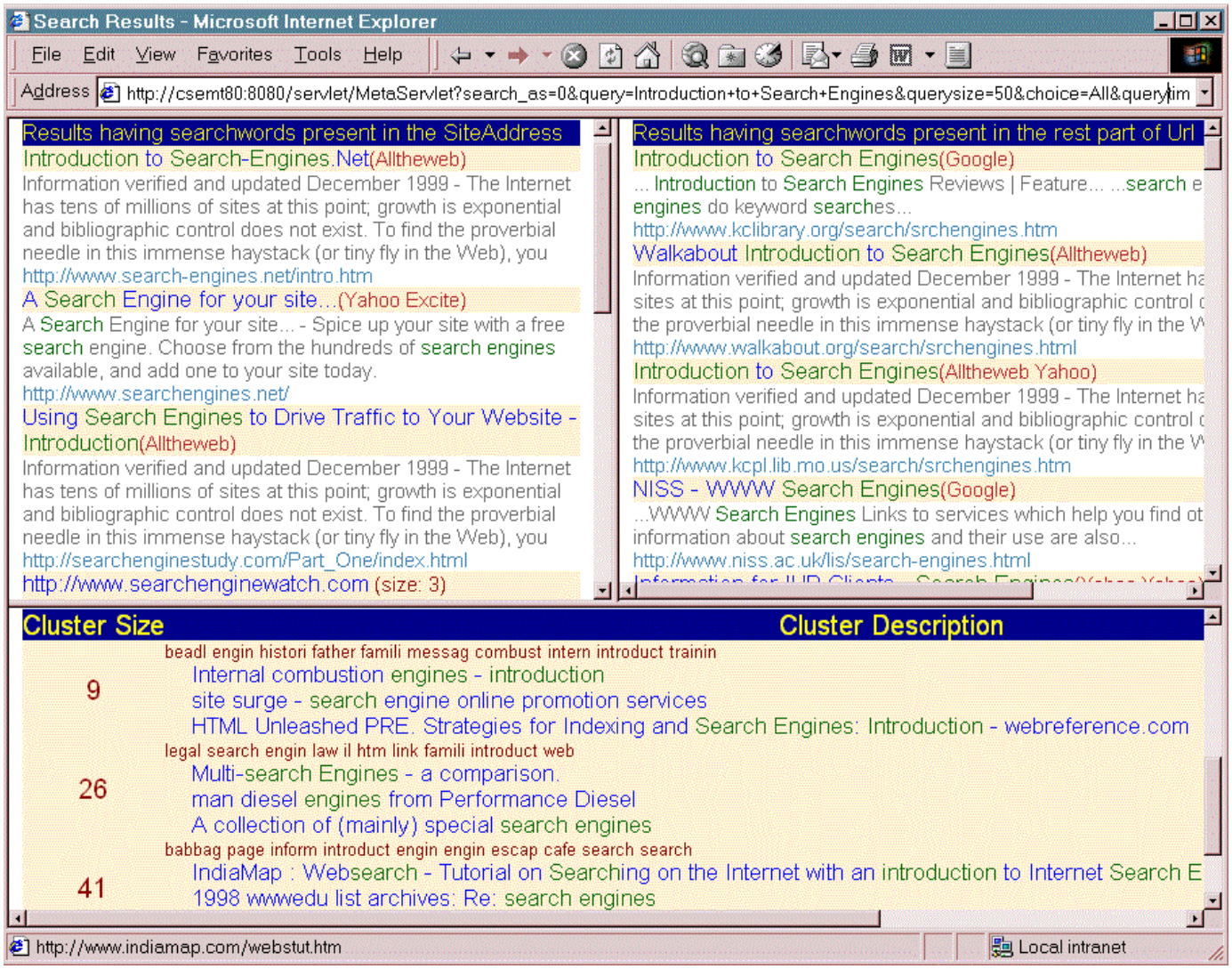


Figure 4.3: Example showing the result for keyword: *Introduction of Search Engines*

Chapter 5

Conclusion and Future Work

This section summarizes the work done in this thesis. We also suggest some possible further work in the area of search engines.

5.1 Experiments

We have attempted to address the hard problem of enabling users to quickly navigate through the results of Web search engines. We have performed some experiments to measure the effectiveness of our search engine. We have tested our search engine on various number of keywords. Results sample are shown below. Simultaneously, we also used other popular search engines for searching documents on the same keywords. The search results are shown on the following pages. From the results we can summarize that:

- *KhojYantra* search results in documents that are either higher in relevance order or equal to those returned by other search engines.
- It also provides clustering of returned documents. For the most relevant and relevant pages, we provide site based clustering so that all the pages returned from a site are listed at one place and the user doesn't have to browse the whole list to find the pages from same site. For general pages, we provide meaning-based clustering. All the pages in their category are divided into a number of clusters on the basis of *keyword-similarity*. In each group it also measure the

relevancy of each cluster with respect to query terms and prioritize the clusters according to their relevancy.

- KhojYantra presents the whole search results into three groups namely, *most relevant*,*relevant* and *general* document groups. Thus by using KhojYantra, the user has flexibility in choosing the documents from the returned document set. He can choose from these categories according to the desired degree of relevance with the keyword.

KhojYantra is a meta-search engine, and it works by retrieving search results from a number of search engines. So, the implementation of *KhojYantra* takes into account the format in which a particular search engine accepts the query-terms and returns its results. If a search engine changes its result presentation or querying format, the particular module for retrieving the data from that search engine must be changed.

5.2 Directions for Further Work

Currently we have implemented only keyword based clustering of documents. We can enhance this by adding provision for concept based clustering. For this we may attach a thesaurus into the search engine that gives the similarity measure for keywords while clustering. This also facilitate the concept based searching, which is performed by retrieving the results for similar words simultaneously.

The additional experiments and extensive studies are necessary to choose the value of constants used in the algorithms used in the system implementation.

At present this meta-search engine performs its technique only on html documents. We can use the same techniques on the xml documents, as they are well structured in nature. These techniques will lead to very good results. Currently this supports the clustering, classification and ranking of documents written in English. We can also apply these techniques to improve the retrieval effectiveness of search engines for other languages, especially Indian Languages.

Search Engines	Top Two Retrieved Search Results
Alltheweb	**Complexity, Complex Systems and Chaos Theory: @Brint.com (tm** ***http://www.brint.com/Systems.htm*** **EPI_System as a theory of uniquely significant systems of over-critical complexity** *http://www.epi-system.unir.ru/*
Infoseek	**Let's Talk Speakers** http://www.webervst.com/sptalk.html **Reason magazine -- December 1997** http://www.reason.com/9712/ed.rick.html
Hotbot	**Adapting science to chaos and complexity theory, chaotic and complex dynamical systems** *http://home.earthlink.net/~imaginationworks/chaos/index.htm* **Primer to Complexity Theory** *http://www.arc.losrios.cc.ca.us/~corcora/compl.html*
Metacrawler	**Complexity, Complex Systems and Chaos Theory: @Brint.com (tm)** *http://www.brint.com/Systems.htm* **MIT LCS Theory of Computation Group** *http://theory.lcs.mit.edu/*
Webcrawler	**MIT LCS Theory of Computation Group** [40] ***http://theory.lcs.mit.edu/*** **IEEE Conference on Computational Complexity** ***http://cs.utep.edu/longpre/complexity.html***
Google	**Complexity, Complex Systems and Chaos Theory: @Brint.com (tm)** *http://www.brint.com/Systems.htm* **Graphs: Theory - Algorithms - Complexity** *http://www.informatik.hu-berlin.de/~weinert/graphs.html*
Grouper	**Computational Complexity Theory** *http://www.cs.umass.edu/~immerman/complexity_theory.html* **Complexity Theory and Theory of Algorithms** *http:/www-cse.ucsd.edu/Research/theory.html*
AltaVista	**The Complexity and Artificial Life Research Concept, CALResCo's educational si** *http://www.calresco.org/* **Algorithms, Complexity Theory, Parallel Computing** http://www.uni-paderborn.de/fachbereich/AG/agma...rste.Seite.html
Yahoo	**Primer to Complexity Theory** *http://www.arc.losrios.cc.ca.us/~corcora/compl.html* **Chaos & Complexity Theory in Education** *http://www.udel.edu/aeracc*
Excite	**Angie's Chaotic World** *http://mainline.brynmawr.edu/~cci98-09/index.html* **Osamu WATANABE's Home Page** ***http://infosys.is.titech.ac.jp/~watanabe/index.html***
KhojYantra	***Most Relevant Pages*** ***http://theory.lcs.mit.edu (size: 16)*** **Theoretical Computer Science -- 1996** **Information and Computation -- 1997** **Complexity theory** ***http://www.complexity-solutions.com/complexity.shtml*** ***Relevant Pages*** **Computational Complexity Theory** ***http://www.cs.umass.edu/~immerman/complexity_theory.html*** **Rice Algorithms & Complexity Group** ***http://www.cs.rice.edu/CS/Algorithms/complexity_theory.html***
	General Pages *complexity theory* *http://www.cultsock.ndirect.co.uk/MUHome/cshtml/introductory/complex.html* **critique of chaos and complexity theory** ***http://platon.ee.duth.gr/data/maillist-archives/sois/1998/msg00174.html***

Figure 5.1: Search Results for the keyword "XML databases"

Search Engines	Top Two Retrieved Search Results
Alltheweb	**Orange-Hill -webdesign, java, javascript, Asp, asp, design, web, databases,dhtml, xml, style** ***http://www.orange-hill.nl/*** **XML Databases - Comparison Products and Services** *http://www.xml-data-servers.com/xml-databases/xml-databases.html*
Infoseek	**TerraServer** http://www.terraserver.microsoft.com/ **Microsoft Music Central** http://www.musiccentral.msn.com/
Hotbot	**XML representation of a relational database** *http://www.w3.org/XML/RDB.html* **Creating High Performance Web Applications using Tcl, DisplayTemplates, XML,** *http://www.binevolve.com/press/pressdocs/tcl-paper/html/tcl.htm*
Metacrawler	**Creating High Performance Web Applications using Tcl, Display Templates, XML** *http://www.binevolve.com/press...ssdocs/tcl-paper/html/tcl.htm* **What is XML?** *http://www.gca.org/whats_xml/default.htm*
Webcrawler	**XML-DBMS, Version 1.0** *http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/xmldbms/xmldbms.htm* **ITO - Staff - Ron Bourret.** *http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/bourret.htm*
Google	**XML and Databases** *www.informatik.tu-darmstadt.de/DVS1/staff/bourret/xml/XMLAndDatabases.htm -* **developerlife.com (tm) - Introduction to XML, Java, databases and the Web** *http://developerlife.com/dbsourceintro/default.htm*
Grouper	**XML-DBMS, Version 1.0** *http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/xmldbms/xmldbms.htm* **ITO - Staff - Ron Bourret** *http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/bourret.htm*
AltaVista	**XML Apps Target Databases, Commerce** *www.techweb.com/wire/story/TWB19980130S0017* **Databases, Tools Push XML Into Enterprise** www.internetwk.com/news1198/news111698-3.htm
Yahoo	**XML.com - Relational Databases.** *http://www.xml.com/pub/Guide/Relational_Databases* **XML, VRML, DHTML, CGI, Perl Scripts, Java, JavaScript, Databases..Ads-Programming.Com.** *http://www.ads-programming.com/web_programming_XML.html*
Excite	**WestLake Internet Training: HTML, FrontPage, Dreamweaver,JavaScript** *http://www.westlake.com/* **RioLabs - All of your buyers, all your suppliers in one place.** ***http://riolabs.com/***
KhojYantra	***Most Relevant Pages*** ***http://www.xmltoday.com(size: 3)*** ***XML Tools: Databases*** ***XML and Databases*** ***http://xml.com(size:2)*** ***XML.com - XML-QL*** ***XML.com-Relational Databases*** ***Relevant Pages*** **DaveNet: XML Databases** ***http://davenet.userland.com/1998/02/22/xmlDatabases*** **Koala XML resources** ***http://www.inria.fr/koala/XML/xml.html***
	General Pages *XML Databases* *http://www.goeast.ch/goeast/archive/0593.html* **VRML, XML, DHTML, HTML, CGI, Perl Scripts, Java, JavaScript,** ***http://www.ads-programming.com/web_programming_VRML.html***

Figure 5.2: Search Results for the keyword "Complexity Theory"

Bibliography

[1] Fast home page. see http://www.alltheweb.com.

[2] Altavista home page. see http://www.altavista.com.

[3] Douglass R. Cutting, David Karger, and Jan Pedersen. Constant interaction-time scatter/gather browsing of very document collections. In *Proceedings of the 16th Annual International ACM/SIGIR Conference*, pages 126–135, Pittsburgh, PA, 1993.

[4] Douglass R. Cutting, David R. Karger, Jan O.Pedersen, and John W. Turkey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In *Proceedings of the 15th Annual International ACM/SIGIR Conference on Research and Development in IR*, June 1992.

[5] Brain S. Everitt. *Cluster Analysis.* Halsted Press New Yark, Toronto, third edition, 1993.

[6] Excite home page. see http://www.excite.com.

[7] G.K.Zipf. *Human Behavior and the Principal of Least Effort.* Addision-Wesley, 1949.

[8] Google home page. see http://www.google.com.

[9] Grouperii see http://zhadum.cs.washington.edu/zamir/grouper2.html.

[10] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *Proceedings of the 19th International ACM*

SIGIR Conference on Research and Development in Information Retrieval, pages 76–84, 1996.

[11] Hotbot home page. see http://www.hotbot.lycos.com.

[12] Infoseek home page. see http://www.infoseek.go.com.

[13] R. Krovetz. Viewing morphology as an inference process. In *In proceedings of the Sixteenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.*, pages 191–202, 1993.

[14] Steve Lawrence and C. Lee Giles. Context and page analysis for improved web search.

[15] Anton V. Leouski and W. Bruce Croft. An evaluation of techniques fro clustering search results. Technical Report IR-76, Department of Computer Science, University of Massachusetts,Amherst, 1996.

[16] Metacrawler home page. see http://www.metacrawler.com.

[17] Northernlight home page. see http://www.northernlight.com/.

[18] M. F. Porter. An algorithm for suffix stripping. *Program*, 14, 1980.

[19] Vijay V. Raghavan and S.K.M.Wong. A Critical Analysis of Vector Space Model for Information Retrieval. *Journal of the Americal Society for Information Science*, 37(2):279–287, sept 1986.

[20] Hinrich Schiitze and Craig Silverstein. Projections for Efficient Document Clustering. In *Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–81, 1997.

[21] See http://www.searchenginewatch.com.

[22] C.J. van Rijsbergen. *Information Retrieval.* London,England,1979, second edition, 1979.

[23] Webcrawler home page. see http://www.webcrawler.com.

[24] Compaq computer coperation. webl home page see http://www.research.digital.com/src/webl/index.html.

[25] Yahoo home page. see http://www.yahoo.com.

[26] Oren Zamir and Oren Etzioni. Grouper: A dynamic clustering interface to web search results.